THESIS

METHODS FOR NETWORK GENERATION AND SPECTRAL FEATURE SELECTION: ESPECIALLY ON GENE EXPRESSION DATA

Submitted by

Nathan Mankovich

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2019

Master's Committee:

Advisor: Michael Kirby

Chris Peterson
Charles Anderson

ABSTRACT

METHODS FOR NETWORK GENERATION AND SPECTRAL FEATURE SELECTION: ESPECIALLY ON GENE EXPRESSION DATA

Feature selection is an essential step in many data analysis pipelines due to its ability to remove unimportant data. We will describe how to realize a data set as a network using correlation, partial correlation, heat kernel and random edge generation methods. Then we lay out how to select features from these networks mainly leveraging the spectrum of the graph Laplacian, adjacency, and supra-adjacency matrices. We frame this work in the context of gene co-expression network analysis and proceed with a brief analysis of a small set of gene expression data for human subjects infected with the flu virus. We are able to distinguish two sets of 14-15 genes which produce two fold SSVM classification accuracies at certain times that are at least as high as classification accuracies done with more than 12,000 genes.

ACKNOWLEDGEMENTS

Thank you to my advisor Michael Kirby for his continued support and patience with me through out this process. A special thanks to Manuchehr Aminian for his software package CALCOM. He also was always available to help answer my questions, debug Python code and provided many links to useful literature. A huge thank you to DARPA for funding this research. Thanks to my parents for their financial and emotional support through undergraduate education and for their continued support of my life decisions. Finally, I'd like to thank my undergraduate advisors Michael Penn and Mike Siddoway for fueling my passion for mathematics and helping me balance my education with my mental health and outdoor pursuits.

DEDICATION

Jason Wells, Chris Vale were two climbers who tragically died in Yosemite in 2018 and 2016.

Their memory reminds me to live every day as if it was my last.

TABLE OF CONTENTS

Chapter 1

Introduction

Feature selection algorithms are prolific in analysis of particularly large and complex data like gene expression data [1]. In this case researchers commonly use feature selection for discovering discriminatory genes for classification problems. These methods can remove irrelevant genes and speed up classification algorithm run-times.

Classic feature selection has three main fields, variable ranking, subset selection and feature construction [1]. An example of variable ranking is an algorithm which ranks features based on their shared information with a desired characteristic [1]. Another ranking method is to employ the F-score as a statistical measure to rank features based on their relevance for an SVM classifier [2]. These ranking methods are limited due to their ranking features based on individual predictive power rather than ranking on the predictive power of subsets. Subset selection aims to solve this problem by ranking subsets of features rather than individual features. An example of a subset selection algorithm is one which approximates the solution to an optimization problem whose objective function is the error of the classifier as a function of the features used [3]. Canonical feature construction algorithms include clustering algorithms [4]. These algorithms group together data based on some measure of similarity. One common clustering algorithm is Laplacian eigenmaps which leverages linear algebraic techniques in conjunction with a heat kernel similarity measure [5]. A special type of clustering is hierarchical clustering which allows for researchers to iteratively builds clusters by either starting with the whole data set as one cluster and iteratively partitioning the data or starting with each data in its own cluster and iteratively combining data points until the whole data set is in one cluster. If a certain cluster size is desired, a researcher can choose to simply take the clusters near this size from the hierarchical clustering process [4]. The wheelhouse for this method is defining some similarity/distance between data points. Our choice for similarity effects our assumption for the proper ambient space in which the data resides.

WGCNA (weighted gene correlation network analysis) uses correlation based measures of distance to perform hierarchical clustering algorithms on gene expression data. The use of this measure makes the assumption that the best space for the data is on a sphere. Therefore, the analysis is limited due to the lack of exploration into other distance measurements. Even with this limitation, this implementation of hierarchical clustering has been able to isolate biologically significant gene feature sets [6].

Our focus for this paper will be to develop multiple methodologies for feature selection on gene expression data using networks in order to determine which features are indicative of a human subject's shedding (contagious) status with the flu virus. In addition to feature selection, we will introduce an unsupervised classification method. Although versions of these algorithms have been explored in various papers, this masters thesis breaks new ground via applying these methods to gene expression data, specifically applying these methodologies to the GSE73072 human influenza challenge dataset.

All of these algorithms are based on realizing our data as a network. We will investigate different methods for network generation which rely on different similarity measures. As stated earlier, different measures of similarity have severe consequences on our assumptions about space in which our data resides. Our first method for feature selection takes the path of iteratively cutting the network into pieces and choosing the 'best' piece as the selected features. This cut is defined using some basic linear algebra techniques but the effectiveness of the cut relies on results from spectral graph theory and optimization problems. The second method relies on providing a ranking system for the genes via two measures of centrality, then tests with the highest ranked genes. We can think of this ranking mechanism being similar to ranking students in a high school classroom based on their popularity. We will be testing our methods using a sparse support vector machine (SSVM) classifier which partitions our data into two sets using a hyperplane. Ideally, these methodologies produce a small number of discriminatory biomarkers along with a classifier that could allow for analysis of a small subset of a single blood sample gene expression profile to distinguish between contagious and healthy subjects. In addition to feature selection for improved SSVM classification,

we will experiment with an unsupervised algorithm which partitions the data into sets using one iteration of the same cutting algorithm used earlier for feature selection.

Chapter 2

Background

2.1 Mathematical Framework

A network (aka graph), $G = (N, E)$, is a collection of nodes $n \in N$ and edges $e \in E$. In this network, each node represents a random variable and each edge represents relationship between two nodes (random variables). An edge weight is a real number that is attached to a given edge. Once a network is generated, we cluster the random variables, or use centrality algorithms to rank the variables from least central to most central in the network. Centrality is a general concept and the definition of centrality varies depending on the algorithm used.

In order to create a network from data we need to determine a way to create edges. Edges should, as accurately as possible, represent the relationship between two nodes in the network. Therefore, our essential question in edge generation is the following: how can we mathematically define the similarity between two variables? The canonical statistics answer to this question is correlation [4]. Researchers have also utilized more complex techniques like partial correlation and mutual information [7], [8]. Some researchers build networks iteratively to ensure a scale free network. Scale free is defined as the distribution of edges approximating the power law distribution with a power between 2 and 3 [9]. However, recent research has begun to debunk this common assumption [10]. Although not commonly used for gene networks, heat kernel is another similarity measure which has been used as the first step in Laplacian eigenmaps [5].

When building these networks we can choose to make edges unweighted or weighted. Thresholding provides a way to convert weighted edges to unweighted edges and is used as a tool to manipulate edge weights in a weighted network. This can also be used to bring a network closer to satisfying the scale free criteria. If the data is a time series, then our goal is to generate a network that represents the evolution of the original network over time which can be modeled by a supra-adjacency matrix [11].

Once a network has been generated, one can begin to analyze the data using the generated network structure. Clustering can be used as a feature selection method that essentially partitions the dataset into subsets that have similar characteristics. This can be used to reduce the dimensionality of the data and pull out characteristic subsets of the entire dataset. Clustering algorithms can be used to create machine learning classification models where we define mapping from the dataset into the various clusters. Also, researchers have applied convolutional neural networks to graphs (supervised learning) [12] as well as hierarchical clustering combined with tree cuts (unsupervised learning) [6]. In addition to standard hierarchical clustering, researchers have done clusterings via spectral methods [13], [5].

A second method for network analysis is centrality. Applying a measure of centrality to a network ranks the nodes from least central to most central. In fact, there are many notions of centrality. A network's most central node can vary depending on the given definition of centrality. Common measures of centrality include k-path, betweenness, and closeness centrality [14,15]. For example, the Krackhard kite graph has different most-central nodes depending on our definition of centrality [16].

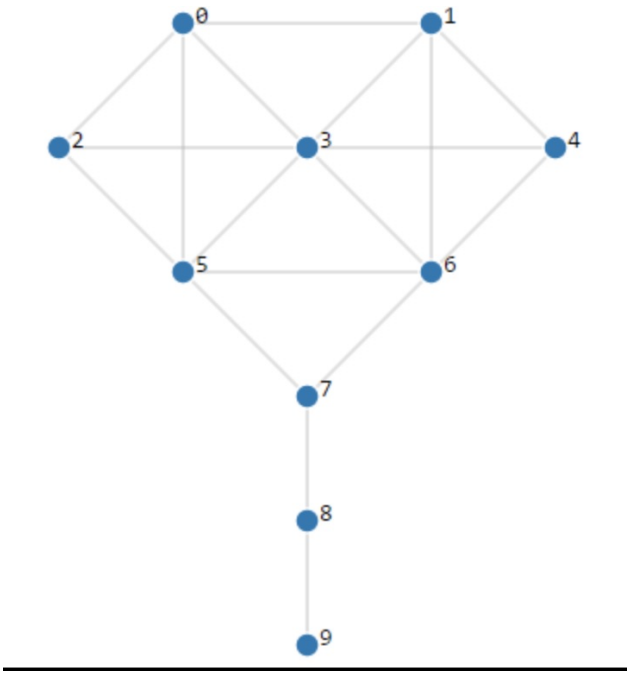


Figure 2.1: The Krackhard kite graph.

With degree centrality node 3 is the most central node. Node 7 is the most central node using betweeness centrality. The most central nodes with closeness centrality are nodes 5 and 6. Researchers have also used certain centrality measures on time-evolving networks which are represented by a supra-adjacency matrix [11,17]. The supra-adjacency matrix stores edge information for the network as it evolves over time and contains extra information for temporal relationship between data.

2.2 Application to Data

Now that we've introduced analysis techniques we will introduce the types of data analyzed by networks. In general, networks are used to analyze any data where the variables in the dataset are "related" in one way or another. For example, pixels in images can be said to be related if they are close together [13]. Even websites are related by their hyperlinks between pages [15]. Our focus for real world application of these algorithms will be microarray gene expression data where the relations between genes is commonly quantified by correlation. When using networks to analyze microarray data, the nodes generally represent genes and these networks are commonly called gene co-expression networks. Many researchers working with microarray stop after generating a network and then compare their results to known network structures in biological data [7,8]. Others generate the networks then perform clustering algorithms (commonly hierarchical clustering) to detect subsets of genes (aka modules) that can be statistically shown to be linked to biological processes [6]. Standout research in gene co-expression networks include the ARACNE algorithm, the PCIT algorithm and the WCGNA package in R.

ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks) is an algorithm for purely network construction, specifically construction of transcriptional regulatory networks. It uses gaussian kernel mutual information estimation to generate edge weights in a network. ARACNE was tested by reconstructing a B cell specific regulatory network. Results were validated by comparing the edges of a known sub-network of the entire network, c-MYC, with biologically known interactions between genes in c-MYC. The researchers noticed that 51.8% of these edges

are also known biological interactions. Also, they found that the genes in their synthetic c-MYC network exhibited some primitive clustering tendencies [7].

The PCIT (Partial Correlation and Information Theory) algorithm improves on the results of the ARACNE algorithm by using a ratio between direct correlation and 'first-order' partial correlation to compute edges. Again, this algorithm is purely used to generate networks and results were compared with known biological interactions between genes. When compared with ARACNE, PCIT generated very different networks with small numbers of genes, but when more genes were introduced, PCIT and ARACNE produced networks with 96% of the same edges [8].

In contrast to ARACNE, the WGCNA (Weighted Gene Co-expression Network Analysis or Weighted Gene Correlation Analysis) package generates a network using correlation and sometimes topological overlap measure. Then it provides a thresholding function to remove edges with small weights. Once a network is generated, this package allows for plotting edge degrees to verify that it satisfies the scale free topology criterion. It also can perform hierarchical clustering using R's built in function. Then the package allows for dynamic tree cutting in order to identify clusters of genes (referred to as modules). Once modules are identified, the package can determine the most significant modules by computing correlation between the biological trait in consideration and either the first principal component of data for each gene in the module (called the eigengene) or the most central gene in the module (called the hub gene). Other features of this package include a looser version of clustering where a gene can be a member of more than one cluster, summary topological statistics, visualizations like dendrograms and smooth interfacing with other software packages. Many of the methods in this paper are more deeply described and tested in [18]. In [6], the aforementioned software package was applied to a mice dataset of liver expression levels. They detected 18 modules in the data and computed correlations between the eigengenes and different physiological traits. One result presented is the correlation between three different modules and the physiological trait, body weight with r-values between 0.43 and 0.59 and p-values between 10^{-7} and 10^{-14} [6].

Although it is common for researchers to use WGCNA to analyze gene expression data, there is little to no canon regarding the use of WGCNA on gene expression data for human flu patients. The dataset in this paper is microarray data from a single pathway that consists of 56 genes and their expression levels for infected human subjects before and during infection with different flu viruses [19]. This paper aims to solve the following problems:

1) Determine a subset of these genes that are the most predictive in determining whether a given subject is a shedder (contagious with the flu) or a control (not infected with the flu). Our tool will be networks where each node in the network represents a gene.

2) Directly classify between shedders and controls at a given time point using all the genes and a network where nodes represent subjects.

To solve the first problem, we present two methods for determining modules. The first method generates a co-expression network, then uses a novel hierarchical clustering algorithm that leverages the eigenvector of either the graph Laplacian or normalized graph Laplacian to determine the modules at single time points. We then feed the genes in these modules to a sparse support vector machine (SSVM) to predict whether a subject is a shedder or a control at single time points. The module with the highest classification rate will win. The second method uses a supra-adjacency matrix to model the data over multiple time points and applies either the largest eigenvector or the PageRank centrality algorithm to determine the best subset of genes. Then we use this the subset of genes to classify between shedders and controls using an SSVM. Both methods provide a way to remove unimportant data from our dataset.

The second problem is essentially an unsupervised classification problem. In order to classify subjects as shedders or controls, we will generate a network where the nodes represent human subjects. We then separate the subjects into two clusters via the same eigenvector method and test to see if these clusters align with the shedder and control labels.

Not only does this paper apply novel spectral clustering algorithms to a new dataset, it also explores different methods for edge generation. For generating each of these different networks, we experiment with unsigned correlation, partial correlation, heat kernel and random.

In summary, this study provides a mathematical foundation for network generation, iterated spectral clustering and centrality analysis, then applies these tools to some small examples and finally to an microarray dataset.

Chapter 3

Methods and Algorithm Development

3.1 Network Generation

Suppose we are given some set of n random variables and p observations of each variable. We create a vector out of the ordered set of observations of a single random variable and call it a *random vector*. The set of n random vectors will be denoted as $\mathcal{X} \subset \mathbb{R}^p$. The goal is to determine the $k < n$ features (subset of random vectors) that contain the most information from the original n variables. To approach this problem we take a network theoretic prospective. Each node in the network represents a random variable and each edge represents a similarity between a pair of random variables. We can construct an adjacency matrix $\mathbf{A}$ for a network via defining the i, jth entry of the matrix as the weight of the edge between node i and node j. We will use the following notation

$$\mathbf{A} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n,1} & w_{n,2} & \dots & w_{n,n} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

The following section explores four different ways of generating edge weights $w_{i,j}$ for this network. These methods can be thought of as maps which take elements from $\mathbb{R}^p \times \mathbb{R}^p$ to $\mathbb{R}$. One note is that we with to not use self-edges in the network. So once we generate edges for the network using the maps below, we force replace whatever is on the diagonal of $\mathbf{A}$ with zeros.

3.1.1 Correlation

Given two random mean centered vectors $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, Pearson correlation is computed as follows:

$$Cor(\mathbf{x}, \mathbf{y}) = \frac{Cov(\mathbf{x}, \mathbf{y})}{\sigma_x \sigma_y} = \frac{E(\mathbf{x})E(\mathbf{y})}{\sigma_x \sigma_y},$$

where $Cov(\mathbf{x}, \mathbf{y})$ is the covariance of $\mathbf{x}$ and $\mathbf{y}$, $E(x)$ is the expected value of $\mathbf{x}$ and σ_x is the standard deviation of $\mathbf{x}$. We can also write this in terms of inner products to gain geometric intuition for this measure of similarity.

$$Cor(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{||x||||y||} = \cos(\theta)$$

θ is the angle between the vectors $\mathbf{x}$ and $\mathbf{y}$. If $\mathbf{x}, \mathbf{y}$ are not mean centered we call the same computation the uncentered correlation. In order to keep the weights in our networks positive we will use unsigned correlation, namely $|Cor(\mathbf{x}, \mathbf{y})|$, for edge weights.

3.1.2 Partial Correlation

Given a pair of random vectors $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, the partial correlation is defined as the correlation between the residuals of $\mathbf{x}$ and $\mathbf{y}$ when projected onto the span of the rest of the data. To begin to solidify this definition in mathematical notation, we will define the rest of the data as $\mathcal{X}' = \mathcal{X} \setminus \{\mathbf{x}, \mathbf{y}\}$. It is common to think of these projections of $\mathbf{x}$ and $\mathbf{y}$ as linear regressions (in fact, least squares regressions) of $\mathbf{x}$ and $\mathbf{y}$ using the dataset $\mathcal{X}'$. We define partial correlation using mathematical notation as follows

$$\text{ParCor}(\mathbf{x}, \mathbf{y}) = \text{Cor}(\mathbf{r}_x, \mathbf{r}_y) = \frac{\langle \mathbf{x} - \hat{\mathbf{x}}, \mathbf{y} - \hat{\mathbf{y}} \rangle}{||\mathbf{x} - \hat{\mathbf{x}}||||\mathbf{y} - \hat{\mathbf{y}}||}$$

where $\mathbf{r}_x$ and $\mathbf{r}_x$ are the residuals of the projections ($\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$) of $\mathbf{x}$ and $\mathbf{y}$ onto span($\mathcal{X}'$) [20].

The common definition for partial correlation uses least squares regression. So in this case the projection matrix is $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ where the columns of $\mathbf{X}$ are formed by the vectors in $\mathcal{X}'$. So in this case we define $\hat{\mathbf{x}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{x}$ and $\hat{\mathbf{y}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

Notice, this is only defined when $\mathbf{x}$ and $\mathbf{y}$ are not in $span(\mathcal{X}')$, otherwise either $||\mathbf{x} - \hat{\mathbf{x}}|| = 0$ and/or $||\mathbf{y} - \hat{\mathbf{y}}|| = 0$.

Suppose we are dealing with p observations of n random variables. If $n >> p$ then we are more likely to have undefined partial correlation because not all n vectors in $\mathbb{R}^p$ can be linearly independent. As a consequence, this method is best used in tall data matrix problems (where

$p > n$). For the sake of using partial correlation to generate edges in networks where $n >> p$ we will use the convention that if $||\mathbf{x} - \hat{\mathbf{x}}|| = 0$ and/or $||\mathbf{y} - \hat{\mathbf{y}}|| = 0$ then ParCor$(\mathbf{x}, \mathbf{y}) = 0$ rather than being undefined.

In summary, partial correlation solves the problem of network-wide interactions influencing the correlation between two nodes. However, partial correlation is limited due to its dependency on having more observations than random variables. In addition, if we are computing partial correlations between all n random variables in a dataset, then we have to do $\frac{1}{2}\binom{n}{2}$ linear regressions which is computationally expensive. Again, in order to keep the weights in our networks positive we will define edge weights using unsigned partial correlation, $|ParCor(\mathbf{x}, \mathbf{y})|$.

3.1.3 Heat Kernel

Heat kernel is used in [13] and [5]. Given two random vectors $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, the similarity between $\mathbf{x}$ and $\mathbf{y}$ is

$$\begin{cases} e^{-\frac{||\mathbf{x}-\mathbf{y}||^2}{\sigma^2}} & \text{if } ||\mathbf{x} - \mathbf{y}|| \leq r \\ 0 & \text{otherwise} \end{cases}$$

where $\sigma \in \mathbb{R}$ is a tunable parameter that is sometimes called the heat kernel and r is another tunable parameter. To simplify this calculation, we take $r = \max_{\mathbf{x},\mathbf{y} \in \mathcal{X}} ||\mathbf{x} - \mathbf{y}|| = 1$ which forces all similarities to be strictly greater than zero. Thresholding will play the same role as r in our similarity calculations.

3.1.4 Thresholding

For each of these methods we can introduce a thresholding parameter which defines a thresholding map from $\mathbb{R}$ to $\mathbb{R}$. We use a soft threshold t_s to remove edges with weights smaller than M. Given a weight for edge $w_{i,j}$ we define a

$$t_s(w_{i,j}) = \begin{cases} 0, w_{i,j} < M \\ w_{i,j}, w_{i,j} \geq M \end{cases}$$

Thresholding also can be used to generate an unweighted network from a weighted one. Given a weight for edge $w_{i,j}$ we define a hard threshold map t_h as

$$t_h(w_{i,j}) = \begin{cases} 0, w_{i,j} < M \\ 1, w_{i,j} \geq M \end{cases}$$

Notice, if we take

$$M := m(\max\{w_{i,j} \in A\} - \min\{w_{i,j} \in A : i \neq j\}) + \min\{w_{i,j} \in A : i \neq j\} \text{ where } m \in (0, 1),$$

we can then discretize $(0, 1)$ into $\{\varepsilon, 2\varepsilon, \ldots, \left\lfloor \frac{1}{\varepsilon} \right\rfloor \varepsilon\}$ for some $\varepsilon > 0$.

Then take $m \in \{\varepsilon, 2\varepsilon, \ldots, \left\lfloor \frac{1}{\varepsilon} \right\rfloor \varepsilon\}$ to test threshold parameters evenly spaced between the smallest nonzero edge value to the largest edge value in order to find the best threshold for the given set of edge weights.

Thresholding can be used to ensure a network satisfies the scale free topology criterion which is defined as the degree distribution in the network being proportional to the power distribution. Specifically

$$p(d(i)|d(i) = k) \propto k^{-a} \text{ for } i = 1, 2, \ldots n$$

where $d(i)$ is the degree of node i and $k > 0$, $p(d(i)|d(i) = k)$ is the probability that a node has degree k and a is generally between 2 and 3. There are a variety of ways to test if this is true. Some researchers use this criteria to build a network by iteratively adding edges. In this study, our goal is to produce the best results possible. So we ignore scale free topology and tune the threshold until we've found the network that produces the best results.

3.2 Partitioning via the Spectrum of the Laplacian

3.2.1 Cutting a Network

Let G be a network with nodes N and edges E where $|N| = n$. Suppose G is a *complete* network. This means there exists an edge between every two nodes in G. Define a *cut* of G to be a partitioning of N into two disjoint subsets N_1, N_2. The graph induced by N_i (namely, $G_i = (N_i, E_i)$) is called a *bank* of the given cut. We construct the graph $G(N_i)$ using all the nodes in N_i and the edges connecting them. The *size* of a cut is a mapping from Y to $\mathbb{R}$ where

$$Y := \{(N_1, N_2) \ : \ A \cup B = N \text{ and } A \cap B = \emptyset\}.$$

So for any two $(N_1, N_2) \in Y$ we can define the size of a cut to be the sum of the weights of the removed edges.

$$\text{cut}(N_1, N_2) = \sum_{w \in E \setminus (E_1 \cup E_2)} w.$$

Define the *minimum cut* of G as

$$\arg\min\{\text{cut}(N_1, N_2) : \ N_1, N_2 \text{ are the two banks of a cut of } G\}$$

We can find the solution to this problem using the Stoer-Wagner algorithm [21]. However, the minimum cut problem favors cuts where $|N_1| << |N_2|$ or vice-versa, therefore we will modify our optimization problem to account for the size of the cut.

There are two ways to construct a cut without the issue described above. One formulation is the *average cut* whose size is

$$\text{Acut}(N_1, N_2) = \frac{\text{cut}(N_1, N_2)}{|N_1|} + \frac{\text{cut}(N_1, N_2)}{|N_2|}$$

Let E_i' be the set of all edges of G connected to nodes in N_i for $i = 1, 2$. The volume of a cut is $\text{Vol}(N_i, N)$ for $i = 1, 2$ is given by

$$\text{Vol}(N_1, N) = \sum_{w \in E_i'} w$$

The size of the *normalized cut* is

$$\text{Ncut}(N_1, N_2) = \frac{\text{cut}(N_1, N_2)}{\text{Vol}(N_1, N)} + \frac{\text{cut}(N_1, N_2)}{\text{Vol}(N_2, N)}$$

Then we can reformulate the minimum cut problem into the minimum average cut problem

$$\arg\min\{\text{Acut}(N_1, N_2) : \ N_1, N_2 \text{ are the two banks of a cut of } G\}$$

or the minimum normalized cut problem

$$\arg\min\{\text{Ncut}(N_1, N_2) : \ N_1, N_2 \text{ are the two banks of a cut of } G\}$$

This formulation favors cuts where the two induced networks have closer to the same number of nodes, that is $|N_i| \approx n/2$ for $i = 1, 2$.

3.2.2 Spectral Partitioning Algorithms

The following outlines an algorithm that provides an approximation to the solution to the optimization problem of finding the average or normalized minimum cut of a connected network G. Define the Laplacian $\mathbf{L}$ of network G with adjacency matrix $\mathbf{A}$ as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \text{ where } \mathbf{D}_{i,j} = \begin{cases} \sum_{i=1}^{n} w_{i,j} & \text{if } i = j \\ 0 & \text{it } i \neq j \end{cases}$$

So $\mathbf{D}_{i,i}$ is the sum of the *incident* (edges connecting to node i) edge weights of node i. In a network with unweighted edges, $\mathbf{D}_{i,i}$ just corresponds to the degree of node i. Assuming the edges in our network are undirected, $\mathbf{L}$ will be symmetric because $\mathbf{A}$ and $\mathbf{D}$ are symmetric. Now we define the normalized graph Laplacian as follows:

$$\hat{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$$

$\hat{\mathbf{L}}$ will be symmetric as a consequence of the symmetry of $\mathbf{L}$. Graph signal processing researchers do an eigenvalue decomposition of $\mathbf{L}$ rather than $\hat{\mathbf{L}}$ [22], while others do the eigenvalues decomposition of $\hat{\mathbf{L}}$ [13] . In order to maintain generality, the following steps can be done with $\mathbf{L}$ or $\hat{\mathbf{L}}$. For ease of notation we will proceed with $\mathbf{L}$.

$$\mathbf{L} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}$$

Since $\mathbf{L}$ is symmetric, we can choose the columns of $\mathbf{V}$ to be mutually orthogonal eigenvectors of $\mathbf{L}$. That is to say, $\mathbf{V}^T = \mathbf{V}^{-1}$. $\mathbf{\Sigma}$ is a diagonal matrix that contains the eigenvalues of $\mathbf{L}$. Another consequence of the symmetric Laplacian is all our eigenvalues are real. In fact, $\mathbf{L}$ is positive semi-definite [23]. Notice $\hat{\mathbf{L}}$ is also positive semi-defnite as an immediate consequence. Therefore all the eigenvalues of $\mathbf{L}$ and $\hat{\mathbf{L}}$ are non-negative. In fact, the geometric multiplicity of the smallest eigenvalue (which is in fact 0) corresponds to the number of connected components of G. Since we assume G is connected, the geometric multiplicity of this eigenvalue is 1.

Suppose $\mathbf{V} = [\mathbf{v}_1|\mathbf{v}_2|\dots|\mathbf{v}_n]$ and the full set of eigenvalues are $\lambda_1, \lambda_2, \dots, \lambda_n$. Take λ to be the smallest non-zero eigenvalue and v to be the corresponding eigenvector to λ. Note: if we did the eigendecomposition of the Laplacian (not normalized), algebraic connectivity of G would be λ and the *Fiedler vector* of G would be $\mathbf{v}$.

$$\mathbf{v} = \mathbf{v}_i \text{ where } i = \arg\min\{\lambda_i : \lambda_i \neq 0\}$$

Recall the $\mathbf{A}_{i,j} = w_{i,j}$ is the edge weight between node i and node j. This assumes some labeling of the nodes of G as $1, 2, \dots, n$. Let us call the ith entry of $\mathbf{v}$, v_i. We can then use the entries in $\mathbf{v}$ as labels for nodes in G where the v_i is the label for vertex i. Assigning values to each node in G is called a *valuation* of G. We use this labeling to define the cut as follows:

$$N_1 = \{i \in N : v_i > 0\} \text{ and } N_2 = \{i \in N : v_i < 0\}$$

It has been shown by Fiedler that using the Laplacian, this cut results in two connected banks [24]. In fact, the Fiedler vector is a real valued solution to the average cut problem [13]. If we are working with the normalized Laplacian, this cut has been shown to be an approximation to the solution to the normalized minimum cut problem [13], [5] . Proof of this fact can be found in Appendix A. We also provide a proof that the Fiedler vector is an approximate solution to the minimum cut problem in Appendix B.

Summary

Given an adjacency matrix **A** for a connected network G, the spectral cut algorithm can be summarized as follows:

1) Build the Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ or the normalized Laplacian $\hat{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$

2) Perform the eigenvalue decomposition of $\mathbf{L}$ or $\hat{\mathbf{L}}$

3) Find $\mathbf{v} = \mathbf{v}_i$ where $i = \text{argmin}\{\lambda_i : \lambda_i \neq 0\}$

4) Cut G using $\mathbf{v}$ as $N_1 = \{x_i : v_i > 0\}$ and $N_2 = \{x_i : v_i < 0\}$ by removing all the edges connecting N_1 to N_2.

3.2.3 Example

Below is an example generated in MatLab of a random complete undirected network with positive edge weights on fifteen nodes. The edges are sampled from the uniform distribution over $(0, 1)$. We use the second smallest eigenvector of the normalized Laplacian, to cut the network once.

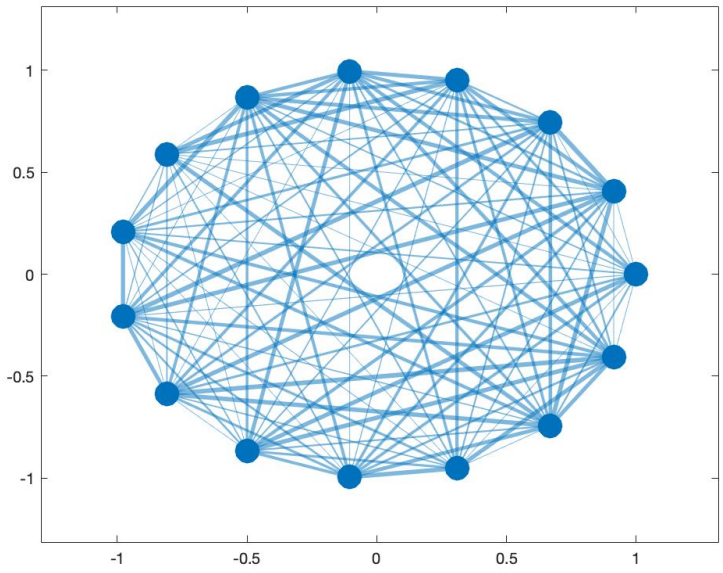


Figure 3.1: The random graph on 15 nodes. The edge thicknesses is proportional to edge weights.

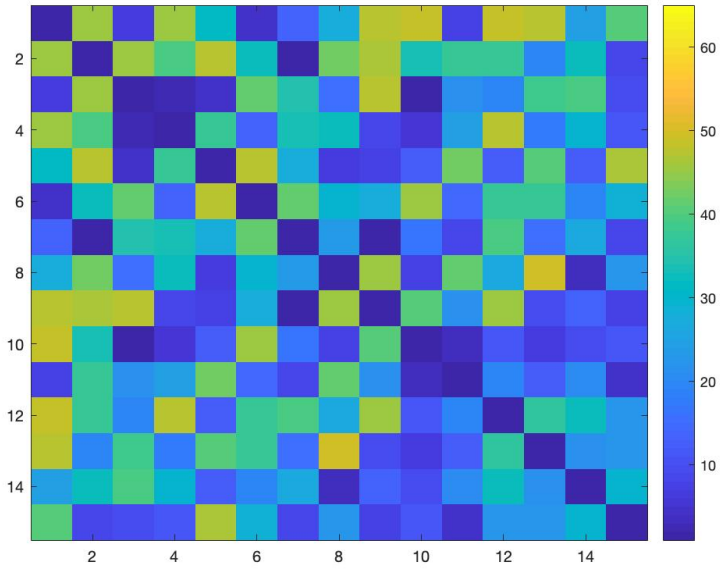


Figure 3.2: A heat map of the adjacency matrix of G scaled by a factor of 50.

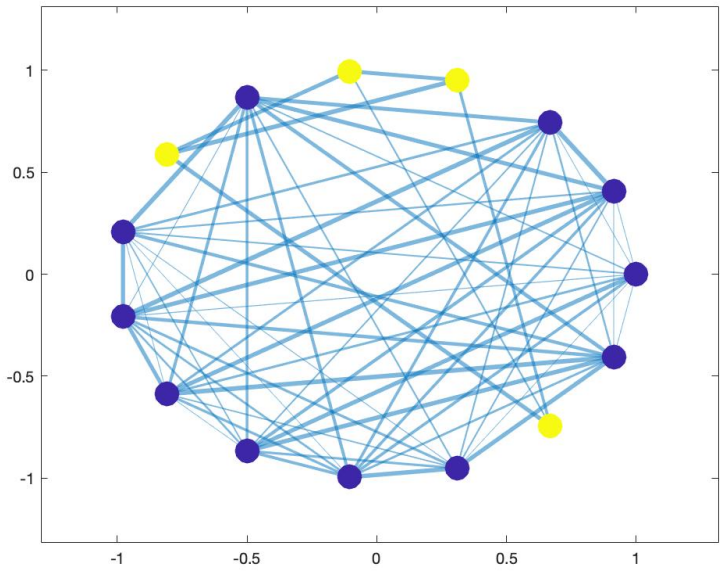


Figure 3.3: The random graph on 15 nodes after spectral cut with the normalized Laplacian. The cut edges are removed. The yellow nodes are in N_1 and the purple ones are in N_2.

Notice one bank size is 4 and the other is11. Even with using an approximation to the minimum normalized cut problem does not guarantee bank sizes of exactly the same size.

3.3 Measures of Centrality and the Supra Adjacency Matrix

3.3.1 Measures of Centrality

Let $G = (N, E)$ be a network with positive edge weights. A centrality is a map $r : N \to \mathbb{R}_{\geq 0}$. The image of this map ranks each node in the network based on its level of centrality. The most central node is defined as the solution to

$$\max_{x \in N} r(x).$$

k-Path Centrality

To begin, let us consider a network G where all the edge weights are 1. For each $x \in N$ we define the degree of x as the sum of incident edges to x. That is,

$$r(x_i) := \deg(x_i) = \sum_{j=1}^{n} w_{i,j}.$$

This is a special case of k-path centrality. To define k-path centrality the centrality score for node x is the number of distinct paths of length at most k beginning at node x. A path is a sequence of adjacent nodes. A path begins at node x if the first node in said sequence is x. Two nodes x_i, x_j are adjacent if they share an edge, eg. $w_{i,j} \neq 0$. So one can say the $\deg(x)$ is the number of paths of length one, beginning x.

A generalization of k-path centrality is eigenvector centrality. Let $\mathbf{A}$ be the adjacency matrix of graph G. Suppose $\mathbf{1} \in \mathbb{R}^n$ is the vector of ones. It is known that the ith entry of $\mathbf{A}^k\mathbf{1}$ gives the number of paths starting at node i of length k [25]. In fact it has been shown in [26] that the eigenvector $\hat{\mathbf{v}}$ of $\mathbf{A}$ which is associated with the largest eigenvalue $\hat{\lambda}$ of A is proportional to the infinite sum of powers of $\mathbf{A}$. In notation:

$$\hat{\mathbf{v}} \propto \sum_{i=1}^{\infty} \frac{1}{\hat{\lambda}^i} \mathbf{A}^i \mathbf{1}.$$

We define the centrality score of node x_i as the ith coordinate of $\hat{\mathbf{v}}$,

$$r_\infty(x_i) = \hat{\mathbf{v}}_i.$$

This is almost the same as the number of paths beginning at x_i of any length, except we are weighting by the inverse of the largest eigenvalue. One could think of this being close to ∞-path centrality. If $\hat{\lambda} < 1$ then this weighting values longer paths. If $\hat{\lambda} > 1$, then this weighting values shorter paths. Finally, if $\hat{\lambda} = 1$ then this is what one might want to define as ∞-path centrality.

In our applications, we are dealing with undirected weighted networks. So paths with larger edge weights result in contributing more to the centrality score than paths with smaller edge weights. This aligns perfectly with our goal of finding the most central node because higher edge weights correspond to higher similarities between our random variables.

Pagerank

Developed in the late 1990s, Pagerank is an algorithm that was originally created as a way to rank web pages on the world wide web. This algorithm is generally used in a directed network, that is, the adjacency matrix is not necessarily symmetric. Let $A \in \mathbb{R}^n_{\geq 0}$ be the adjacency matrix for our network. We generate a transition probability matrix $\mathbf{M}$ whose entries are defined as

$$\mathbf{M}_{i,j} = \frac{\mathbf{A}_{i,j}}{\sum_{j=i}^{n} \mathbf{A}_{i,j}}.$$

The intuition is as follows. Suppose a person is surfing webpages. Each webpage is represented by a node in the network. $\mathbf{M}_{i,j}$ represents the probability that the surfer will travel from page i to page j. We generate the vector $\mathbf{R}^t$ whose ith entry represents the probability the surfer will end on page i after t iterations of the following algorithm. We can initialize this vector in a few ways: either initialize this vector as the uniform distribution $\mathbf{R}^0 = \frac{1}{n}$, sample the coordinates of $\mathbf{R}^0$ from the standard normal distribution, or via another probability distribution depending on what the network is modeling. Then we update as follows

$$\mathbf{R}^{t+1} = d\mathbf{M}\mathbf{R}^t + \frac{1-d}{n}\mathbf{1}$$

where d is a dampening factor. We iterate until the change in the update drops below ϵ,

$$|\mathbf{R}^{t+1} - \mathbf{R}^t| < \epsilon.$$

These parameters can be tuned depending on the given network and desired processing time. Each coordinate in $\mathbf{R}_i$ is the rank of page i. So upon completion of the algorithm at time T we define the rank of node x_i as,

$$r(x_i) = \mathbf{R}_i^T$$

[15]. For this paper we will use $d = 0.85$ and $\epsilon = 0.001$. We set a high damping factor in order to encourage low random exploration for the surfer [**?**, **?**]. Epsilon was chosen arbitrarily as a small

value. To increase algorithm performance one may choose to tune these parameters in a more precise manner.

3.3.2 The Supra Adjacency Matrix

Suppose we are given a time series of networks on the same set of nodes. That is, we have $(G_1, G_2, \ldots, G_T)$ where $G_t = (N, E_t)$ for $t = 1, 2, \ldots, T$. Each G_t has an adjacency matrix $\mathbf{A}_t$ for $t = 1, 2, \ldots T$. Our goal is to realize this series of the same nodes as one large network, where there are T different nodes representing each $x \in N$. That is to say, $x \in N$ is represented by $x_1, x_2, \ldots x_T$. Then we can choose how to draw the edges between each of $\{x_1, x_2, \ldots x_T\}$. The most basic suggestion would be to connect each x_t to each x_{t+1} with a directed edge. Now we have a directed network on $n \cdot T$ nodes. The supra-adjacency matrix $\mathbf{S}$ is the adjacency matrix of this network. Following these steps results in the construction of a matrix of the block form:

$$\mathbf{S} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{W}_{1,2} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{W}_{2,3} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_3 & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{A}_T \end{bmatrix}.$$

$\mathbf{A}_t$ is the adjacency matrix for network G_t. $\mathbf{W}_{t,t+1}$ is the matrix which links the nodes between different time steps, specifically between the network at time t to the network $t + 1$. Each $\mathbf{W}_{t,t+1}$ is a diagonal matrix because we only wish to have edges between nodes to themselves at different time steps [11].

By using this construction we are making an assumption about our network and how it evolves over time. We are assuming there is no direct interaction between node x at time t and any other nodes at any other times except node x at time $t + 1$. Our choice of methodology for construction of the edges between nodes at time t and $t + 1$ can greatly effect the performance of this model under different network algorithms.

3.4 Specialized Algorithms

3.4.1 Iterated Spectral Partitioning

Suppose we are given a network $G = (N, E)$, let m' be the minimum desired cluster size.

1) Perform spectral cut algorithm to cut G into G_1, G_2

2) Repeat step 1 on G_1 and G_2. Stop when the $|N_1| < m'$ or $|N_2| < m'$.

This results in a set of sub networks $\{G_1, G_2, \ldots, G_r\}$ where

$$\left| \bigcup_{i=1}^{r} N_i \right| = |N|$$

Suppose we run this algorithm until all the resulting sub-networks contain only one node. Then we can visualize this algorithm using a dendrogram. A dendrogram is a standard visualization of hierarchical clustering. They are commonly used for visualizing hierarchical clustering for gene expression data and can be generated using numerous different methods for measuring distances between clusters. Suppose G was cut into G_1 and G_2 then the distance between G_1 and G_2 is

$$\text{dist}(G_1, G_2) = \ln(|N_1||N_2|)$$

If G is connected, then each of G_1 and G_2 are connected so $|N_1||N_2|$ is the number of cut edges. This is because the graph defined using all the nodes in N and the cut edges is exactly the complete bipartite graph K_{N_1,N_2}.

3.4.2 Supra-adjacency Centrality

Suppose we are given a time series of networks on the same set of nodes. That is, we have $(G_1, G_2, \ldots, G_T)$ where $G_t = (N, E_t)$ for $t = 1, 2, \ldots, T$.

1) Generate a supra-adjacency matrix $\mathbf{S}$ with the graph G_S.

2) Use a centrality method to generate centrality scores for each node in G_S

3) Generate a centrality for each node x in the network as the sum of the instances of that node in G_S.

$$r(x) = \sum_{t=1}^{T} r(x_t)$$

Again, the choice of the centrality algorithm depends on what the network is modeling and the desired computation time.

Chapter 4

Application to Data

4.1 Synthetic Data

The following is a comparison of edge generation methods by testing the various tools described in the methods section on synthetic datasets. Recall, we use unsigned correlation and partial correlation in order to generate the edges in these networks. Our primary reason for this choice is to use approximations to solutions for types of minimum cuts to partition our data. When we refer to unsigned correlation and unsigned partial correlation as correlation and partial correlation for the remainder of this thesis. Implementation of the aforementioned algorithms for these examples and subsequent real-world data is in Python using the packages numpy, scipy, sklearn and networkx [27], [28], [29], [30].

4.1.1 Edge Generation and Single Cut

First, let us consider an example where partial correlation is able to determine linear dependence of variables while other edge generation cannot. We will always be classifying with the Fiedler vector. Take $n = 15$ random variables with $p = 100$ observations for each variable, $\{\mathbf{x}_0, \ldots, \mathbf{x}_{14}\} \subset \mathbb{R}^{100}$. $A = \{\mathbf{x}_0, \ldots, \mathbf{x}_4\}$ are close together using Euclidean distance and $B = \{\mathbf{x}_5, \ldots, \mathbf{x}_9\}$ are close together using Euclidean distance. Then we define each of $B' = \mathbf{x}_i \in \{\mathbf{x}_{10}, \ldots, \mathbf{x}_{14}\}$ as linear combinations of vectors in B. In the figure below A is orange, B is blue, and B' is green.

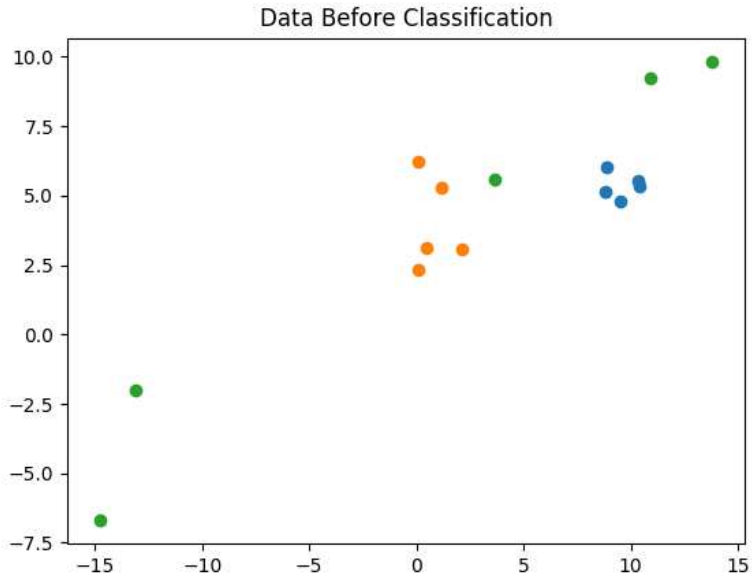
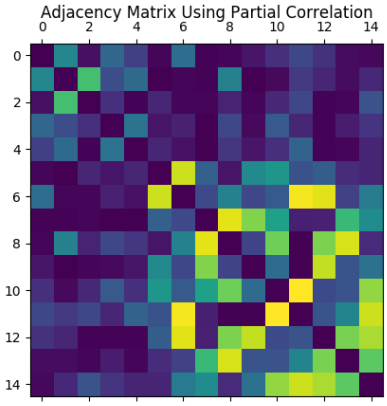
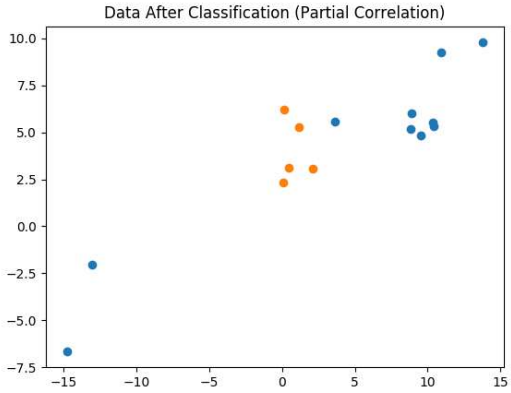


Figure 4.1: A projection of the first two coordinates of each $\mathbf{x}_i$. A is orange, B is blue, and B' is green.

We then generate a network from these data with 15 nodes using partial correlation, correlation, and heat kernel. Then we cut each network using the Fiedler vector. The data that corresponds to the nodes in one bank are blue and the data that corresponds to the nodes in the other bank are orange. Notice this method is able to place the nodes corresponding to the linearly dependent data in one bank and the rest in its own bank.

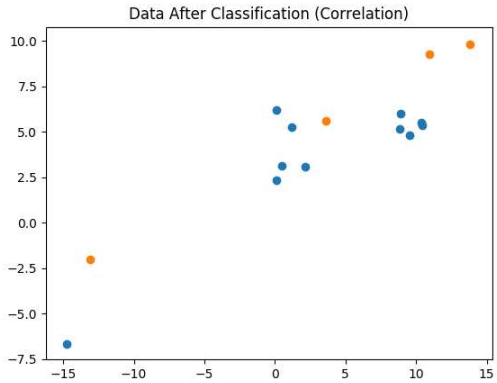


(a) The adjacency matrix.



(b) A projection of the first two coordinates of each $\mathbf{x}_i$.

Figure 4.2: When we use partial correlation to generate the edges of our network our data separates with the nodes representing $B \cup B'$ in one bank and the nodes representing A in the other.



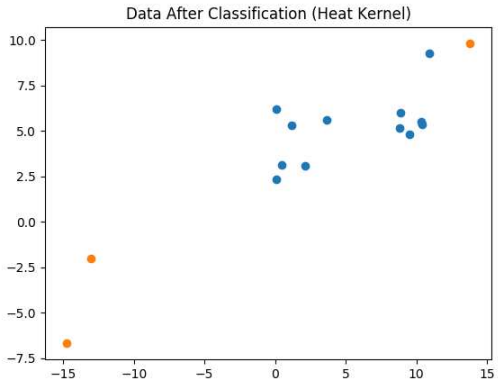


Figure 4.3: These are both projections of the first two coordinates of each $\mathbf{x}_i$. Correlation and heat kernel edge generation produce networks that do not always cut into $B \cup B'$ and A.

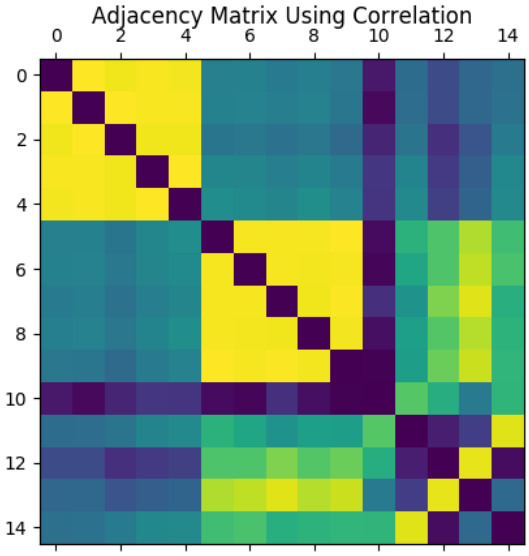
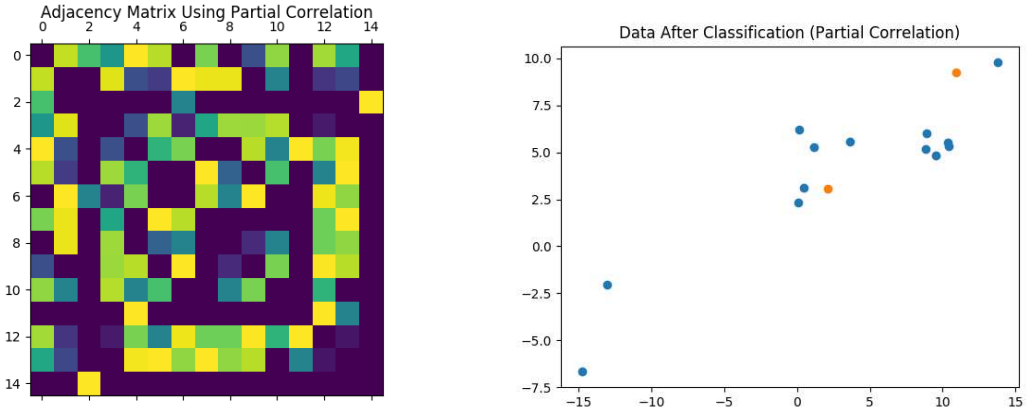


Figure 4.4: Correlation does seem to mildly detect the linear independence in its adjacency matrix.

If we drastically decrease our number of observations, $n = 15$ and $p = 2$, then we can no longer use partial correlation to detect the linear dependence of each vector in B' on the the vectors in B because we are in the situation $n >> p$. Therefore, there will be many other linear dependencies between vectors in our dataset other than just the dependency between vectors in B and B'. As a result we are guaranteed to have 0 residuals of some of our projections which will cause partial correlations of zero. For the figure below, the data that corresponds to the nodes in one bank are blue and the data that corresponds to the nodes in the other bank are orange.

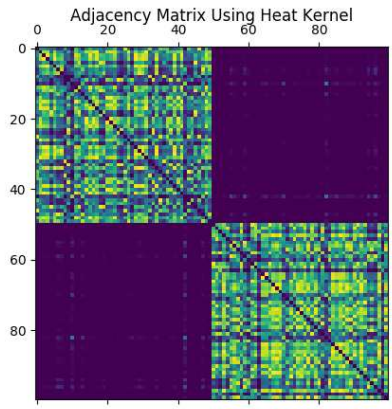


(a) The adjacency matrix. Some of the entries are zero.

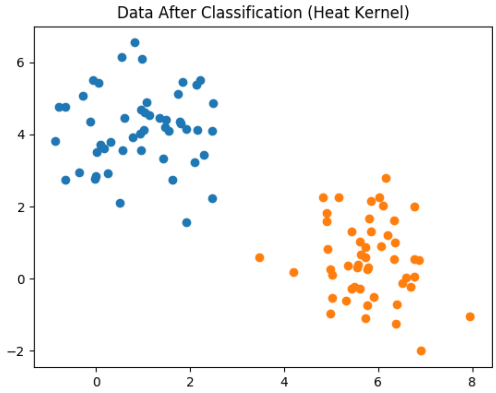
(b) A projection of the first two coordinates of each $\mathbf{x}_i$.

Figure 4.5: When $\mathbf{x}_i \in \mathbb{R}^2$ partial correlation no longer can detect the intended linear dependencies.

If we consider $n = 100$ and $p = 2$ with two clusters (via Euclidean distance) then heat kernel can detect these clusters, correlation misclassifies some points and partial correlation cannot detect anything because $n >> p$. Below is plots of the adjacency matrices and the cut outputs from each edge generation method.

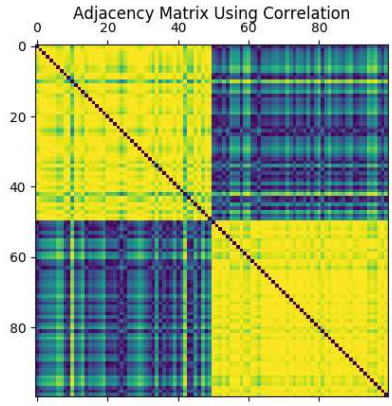


(a) Adjacency matrix.

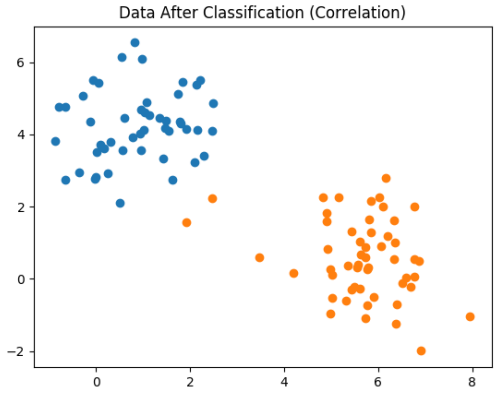


(b) A projection of the first two coordinates of each $\mathbf{x}_i$.

Figure 4.6: Heat kernel edge generation. The data are perfectly separated.

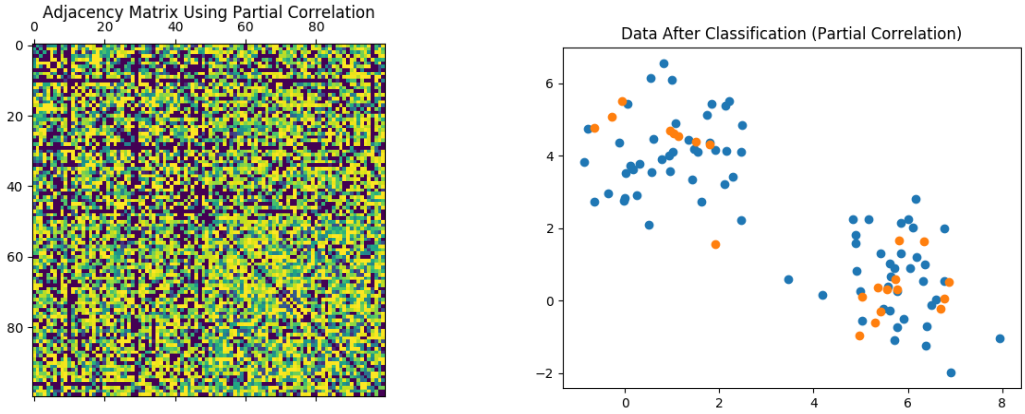


(a) Adjacency matrix.



(b) A projection of the first two coordinates of each $\mathbf{x}_i$.

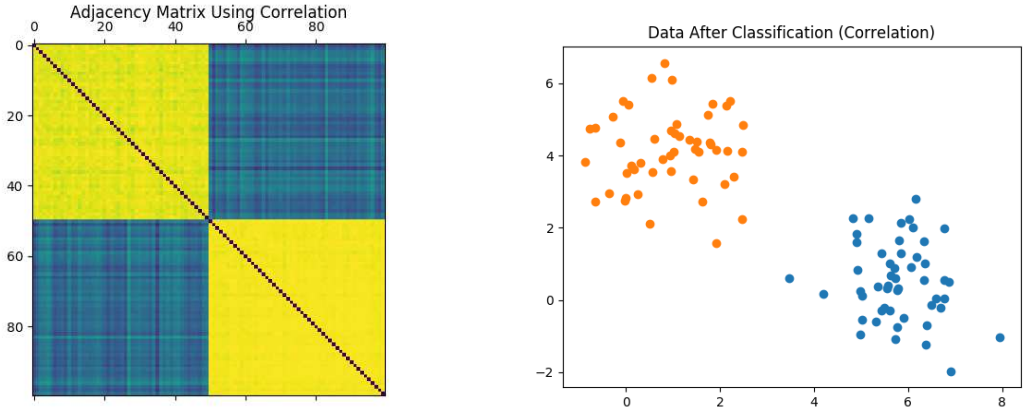
Figure 4.7: Correlation edge generation. The data are not perfectly separated into the two intended clusters.



(a) Adjacency matrix. **(b)** A projection of the first two coordinates of each $\mathbf{x}_i$.

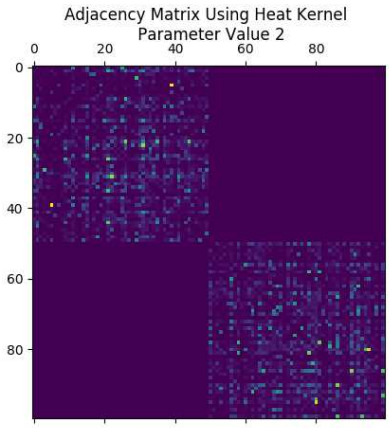
Figure 4.8: Partial correlation edge generation. The data are not separated into the two intended clusters.

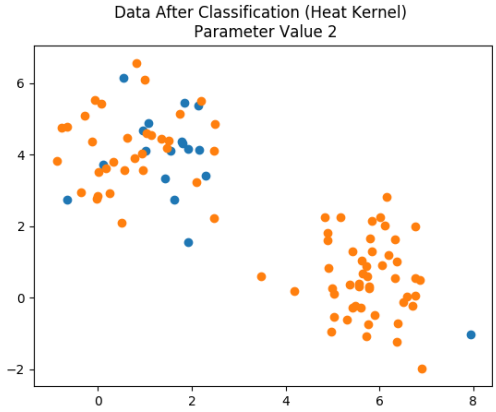
If we let $p = 10$, then correlation classifies perfectly, heat kernel classifies second best and partial correlation still fails due to the relative sizes of n and p. However, if we tune the heat kernel parameter, we are able to classify perfectly with heat kernel. The following figures are projections of the first two coordinates of the dataset.



(a) Adjacency matrix. **(b)** A projection of the first two coordinates of each $\mathbf{x}_i$.

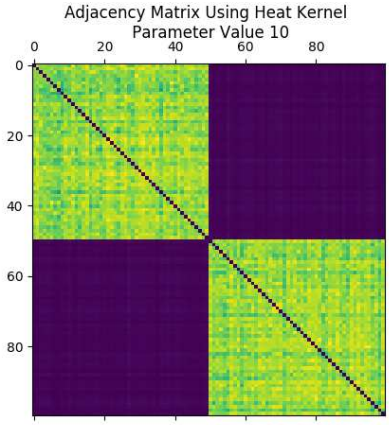
Figure 4.9: Correlation edge generation. The data are perfectly separated.

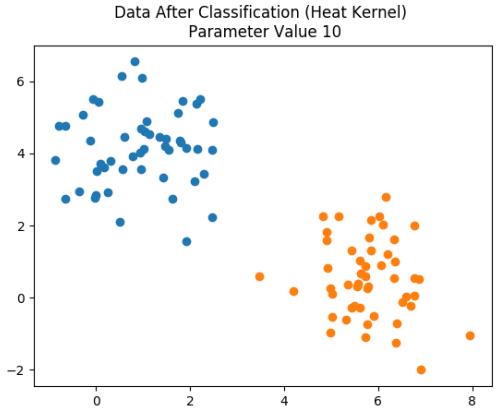




(a) Adjacency matrix. **(b)** A projection of the first two coordinates of each $\mathbf{x}_i$.

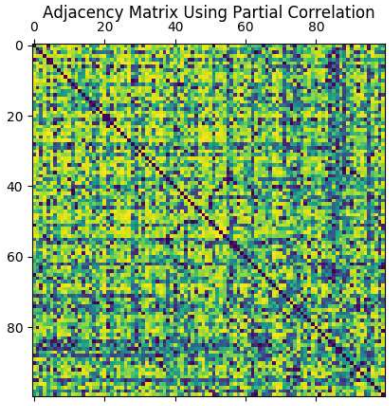
Figure 4.10: Heat kernel edge generation with $\sigma = 2$. The data are not perfectly separated.



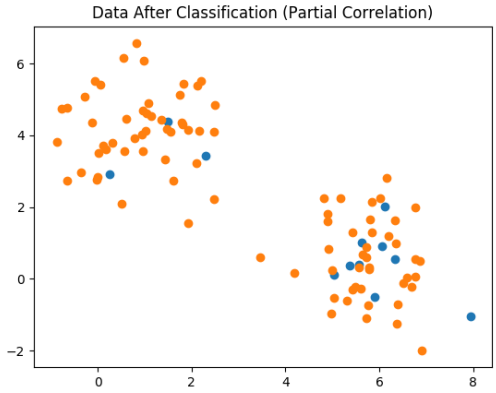


(a) Adjacency matrix. **(b)** A projection of the first two coordinates of each $\mathbf{x}_i$.

Figure 4.11: Heat kernel edge generation with $\sigma = 10$. The data are perfectly separated.



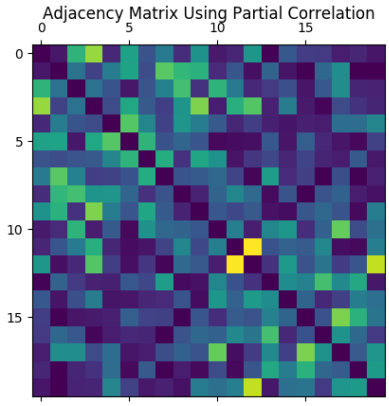
(a) Adjacency matrix.



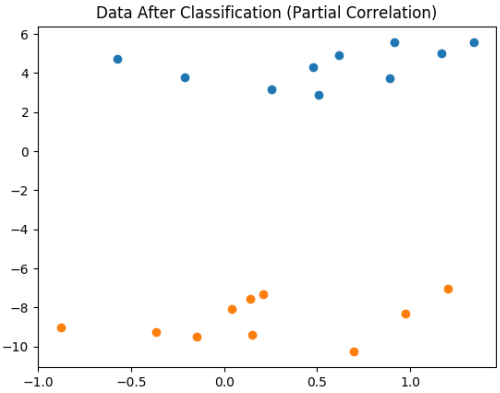
(b) A projection of the first two coordinates of each $\mathbf{x}_i$.

Figure 4.12: Partial Correlation edge generation. The data is not separated into the two intended clusters.

If we let $p = 170$ and $n = 20$, we are able to classify perfectly with partial correlation. If we decrease p, our classification accuracy decreases quickly. Below is the adjacency matrix of the data and a projection of the first two coordinates of the data an the classification using partial correlation to generate the network.



(a) Adjacency matrix.



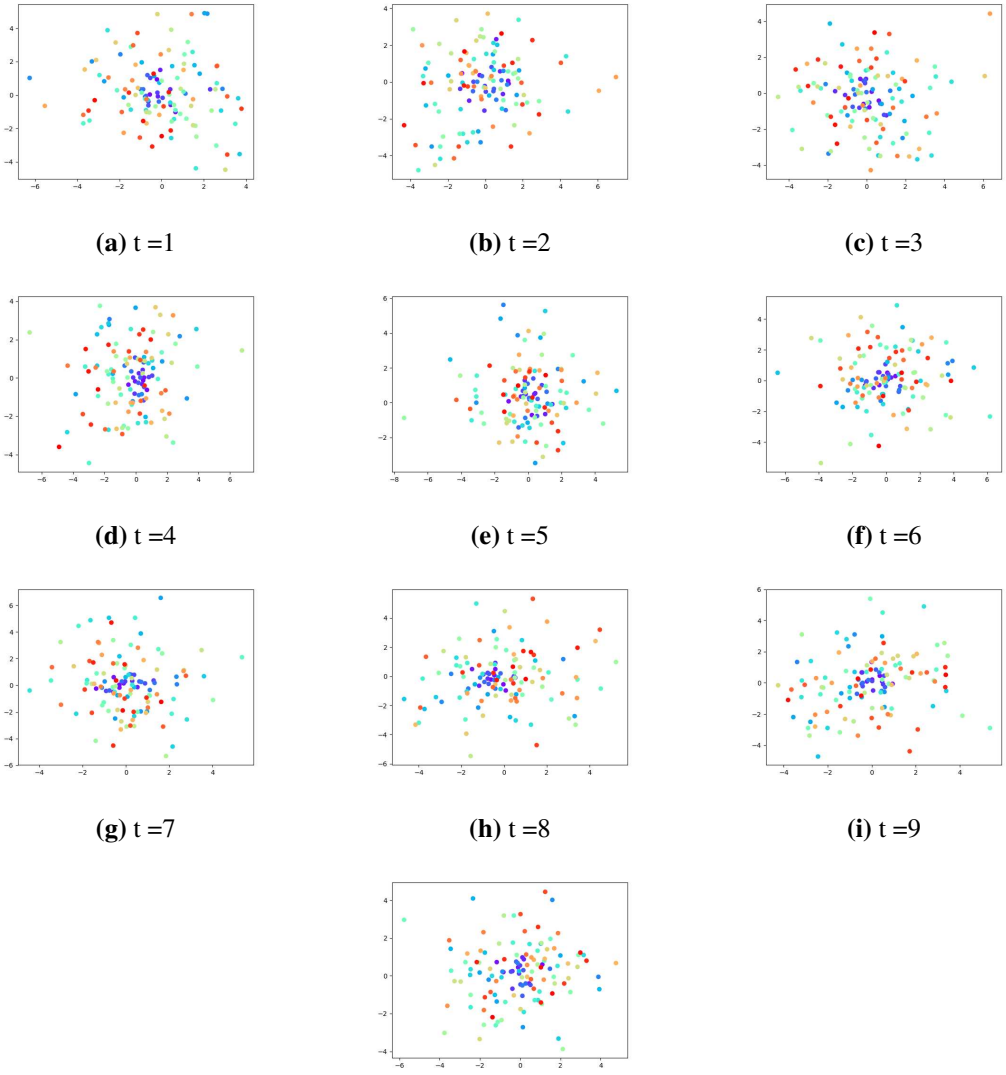
(b) A projection of the first two coordinates of each $\mathbf{x}_i$.

Figure 4.13: Partial Correlation edge generation. The data is separated into the two intended clusters.

4.1.2 Supra-Adjacency Matrix

Now we will have a point cloud with $n = 122$ points in $\mathbb{R}^2$ that evolve over $T = 10$ time steps. Let us denote the points at time t to be $\{\mathbf{x}_1^t, \ldots, \mathbf{x}_{122}^t\} \subset \mathbb{R}^2$. So we have $p = 2$ and $t \in \{1, 2, \ldots 10\}$. The data has been constructed so there is always a subset of 20 points that are generally in the center of the data via Euclidean distance.

We generate a supra-adjacency matrix of the data using heat kernel with $\sigma = 2$ and fix the similarity times steps being fixed at 0.5. Then we use largest eigenvector centrality and PageRank centrality on each time step to find the $k = 20$ most central data points. We choose to use heat kernel because it is a Euclidean distance-based similarity measure and the features that we wish to select remain close to the center of the data with respect to Euclidean distance.



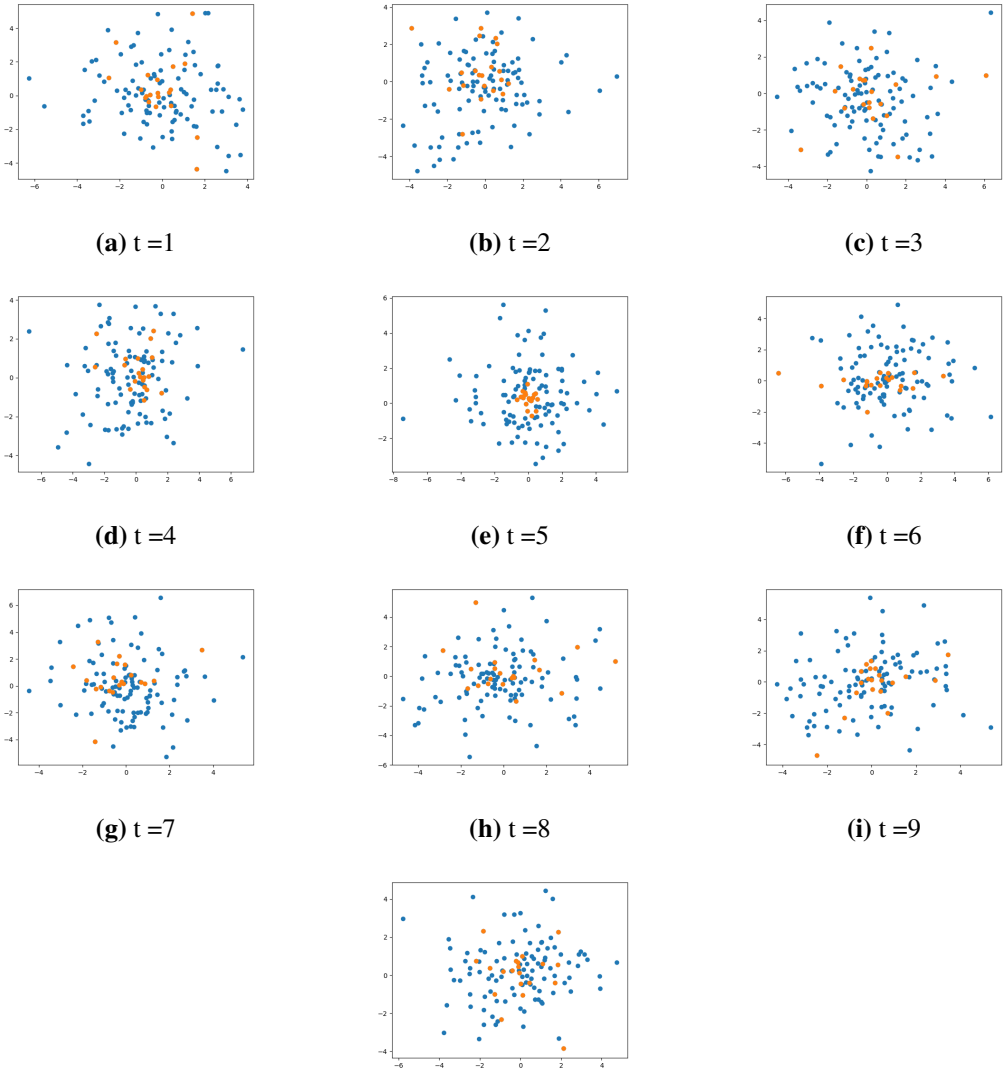
(a) t =1 **(b)** t =2 **(c)** t =3

(d) t =4 **(e)** t =5 **(f)** t =6

(g) t =7 **(h)** t =8 **(i)** t =9

(j) t =10

Figure 4.14: All the data evolving over time. Each color corresponds to a specific data point. Notice the dark purple points generally stay in the center of the data.



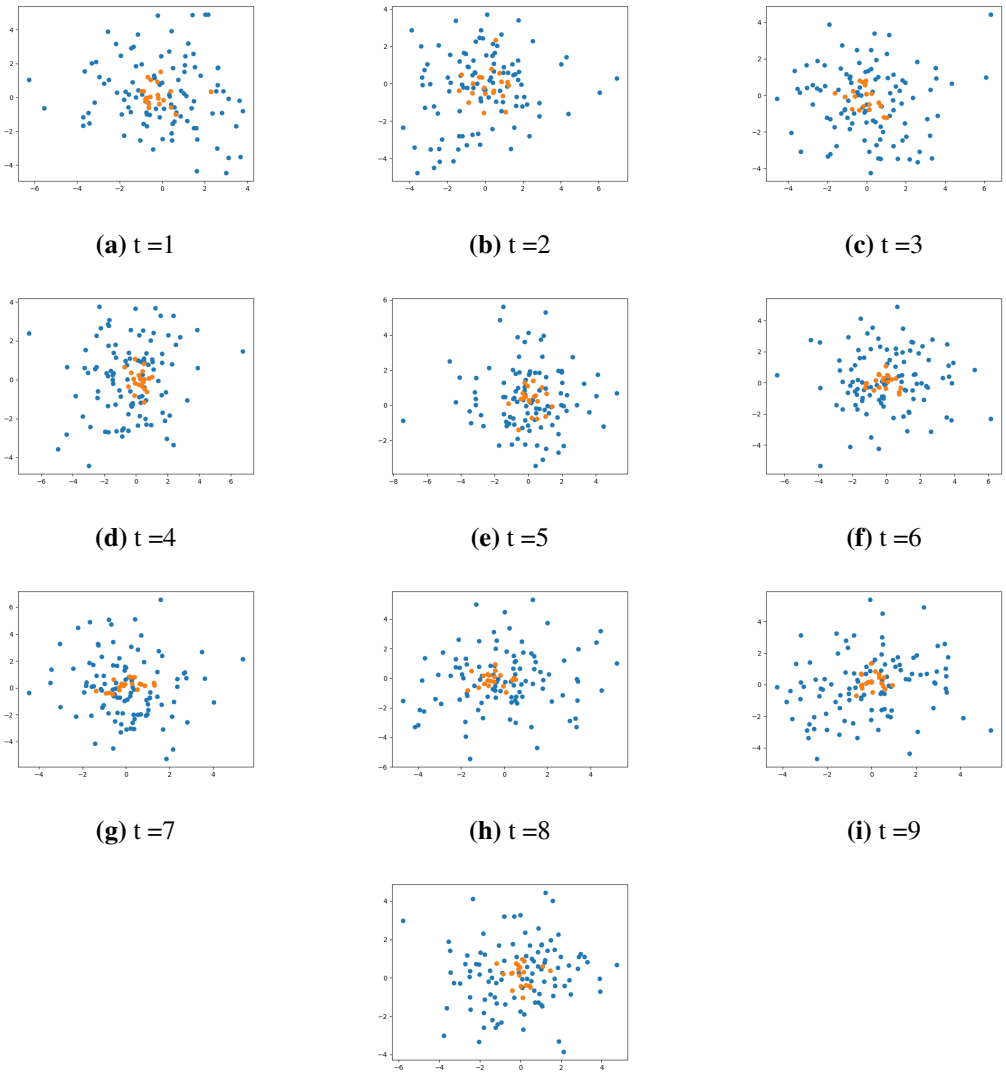
(a) t =1 **(b)** t =2 **(c)** t =3

(d) t =4 **(e)** t =5 **(f)** t =6

(g) t =7 **(h)** t =8 **(i)** t =9

(j) t =10

Figure 4.15: Largest eigenvector centrality $k = 20$ on a heat kernel generated supra-adjacency matrix. Notice this method identified the most central genes to be the genes at $t = 5$ rather than the most central genes overall.



(a) t =1 **(b)** t =2 **(c)** t =3

(d) t =4 **(e)** t =5 **(f)** t =6

(g) t =7 **(h)** t =8 **(i)** t =9

(j) t =10

Figure 4.16: PageRank centrality $k = 20$ on a heat kernel generated supra-adjacency matrix. Notice PageRank out performed largest eigenvector centrality.

Each sub-network corresponding to a time in the network induced by the supra-adjacency matrix is fully connected. Edges between these sub-networks only connect nodes representing the same data point. As a path based centrality measure, largest eigenvector centrality put too much emphasis on the time connections between these sub-networks which resulted in the nodes in the absolute center of the network of the super-adjacency matrix having the highest centrality score. PageRank did not run into this issue since it is more about diffusion of information via transition

probabilities rather than counting paths (although the two measures are both spectral centrality measures).

4.2 Gene Expression Data Analysis

Our data are from the reactome interferon alpha beta signaling pathway from the GSE73072 dataset [19]. This dataset is of the microarray variety. Specifically, the data was collected from 7 studies by Duke, UVA and hVIVO funded by the Defense Advanced Research Projects Agency (DARPA). The entire dataset consists of 12,023 genes and 148 human subjects infected with 4 different strains of the flu virus. These four strains of the flu virus are HRV, RSV, H1N1 and H3N2. Gene expression data for these subjects were collected before and at times up to 680 hours after infection. Preliminary analyses on these data suggested that the genes in the reactome interferon alpha beta signaling pathway carried the same amount of signal for a subject's shedding status as all the 12,023 genes at later time bins. So we will run our algorithms on just the 56 genes in this pathway. Notice the classification accuracies for the pathway genes are low for the first few time bins. So we will choose to do most most of our analysis using data from later time bins.

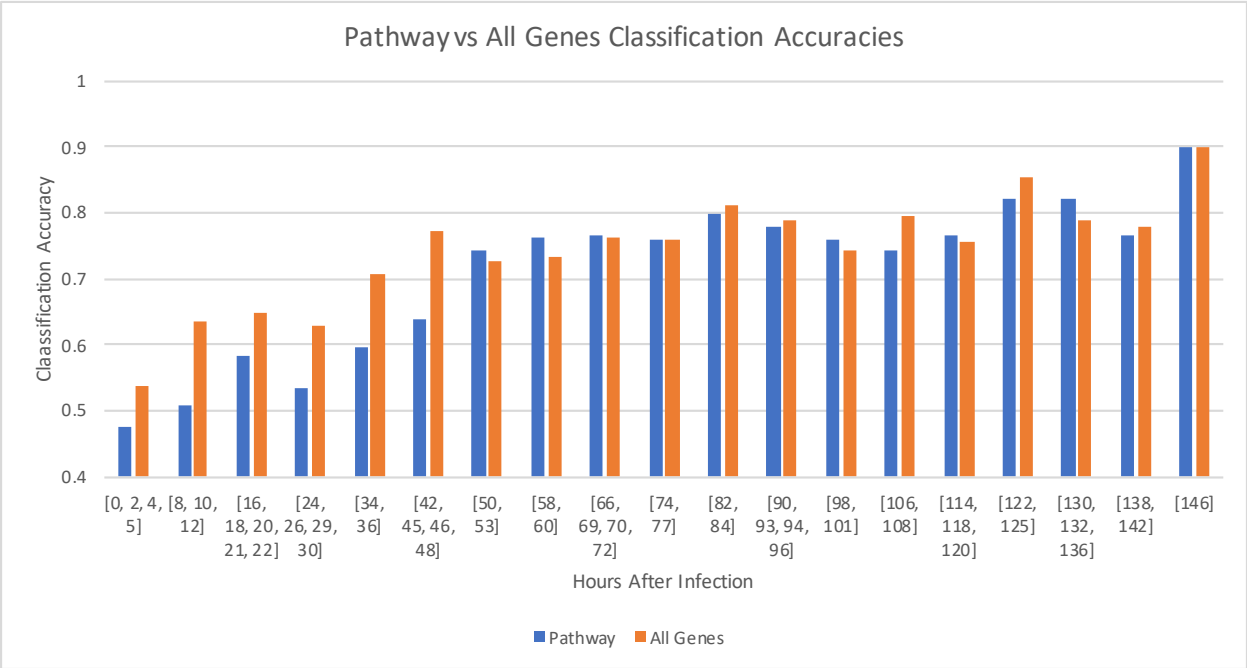


Figure 4.17: 2 fold SSVM classification accuracy for time bins with at most 6 hours of consecutive times. .

In addition, choosing to reduce our dataset size allows us to experiment with different varieties of algorithms and tune parameters in a finer fashion. We define our control dataset to be the pre-infection data (negative times). Then we compare the gene expression levels from the control dataset with the data for shedders at times 0 to 680 hours after infection. The numbers of shedders (infected and contagious subjects) is anywhere between 10 to 81 depend on the time when the data is collected. For all of the proceeding analysis we will pre-process all data via a natural log transformation. For thresholding we will use t_s as described in the methods section. We will choose M using m as described in the earlier section. To find the best m we fix an $\varepsilon = .1$. Then we sample m from the multiples of ε from 1 to 10. We threshold for all methods other than the supra-adjacency centrality algorithm. For all SSVM classifications we will ensure there are the same number of shedders as controls and we will be using a two fold SSVM classifier. Our first goal is to identify a subset of characteristic genes which classify between shedders and controls at least as well as the pathway genes using networks with genes as nodes. Our second goal is to classify between controls and shedders using subjects as nodes.

4.2.1 Genes as Nodes: The Characteristic Gene Problem

1) Iterated Spectral Partitioning Algorithm

We use the simple ISPA with the normalized graph Laplacian to visualize our data in the form of a dendrogram. A dendrogram is a standard tool for visualizing hierarchical clustering algorithms. In the figures below, the structure to the left of the heat maps is the dendrogram. This is essentially a hierarchical clustering tree. In this case, leaves of the tree correspond to individual genes. In the figure below, the leaves are the furthest right lines. If we move our vision along these lines from right to left we can see that these lines begin to join together and eventually all join together at the far left size of the dendrogram. The joining of these lines represents the merging of two clusters. The distance from the singleton cluster to the merging of the clusters is the value of the distance between the two clusters using the chosen clustering algorithm's distance definition. Since for our application the relative distances between clusters are more important than the actual distance

value, we choose to omit labels for the distances axes in these plots. Depending on our clustering algorithm, it can be better to read left to right or right to left along these dendrograms. In the case of the ISPA algorithm, the dendrogram is generated from left to right since we cluster by beginning with the entire dataset as one cluster and recursively cutting the dataset into smaller pieces until we are left with all clusters of size 1. The heat maps to the right of the dendrograms are visualizations of the raw gene expression data for a given subject. Genes are on the vertical axis and subjects are on the horizontal axis. Below is a comparison of four dendrograms generated using shedder data at time 60 hr after infection.

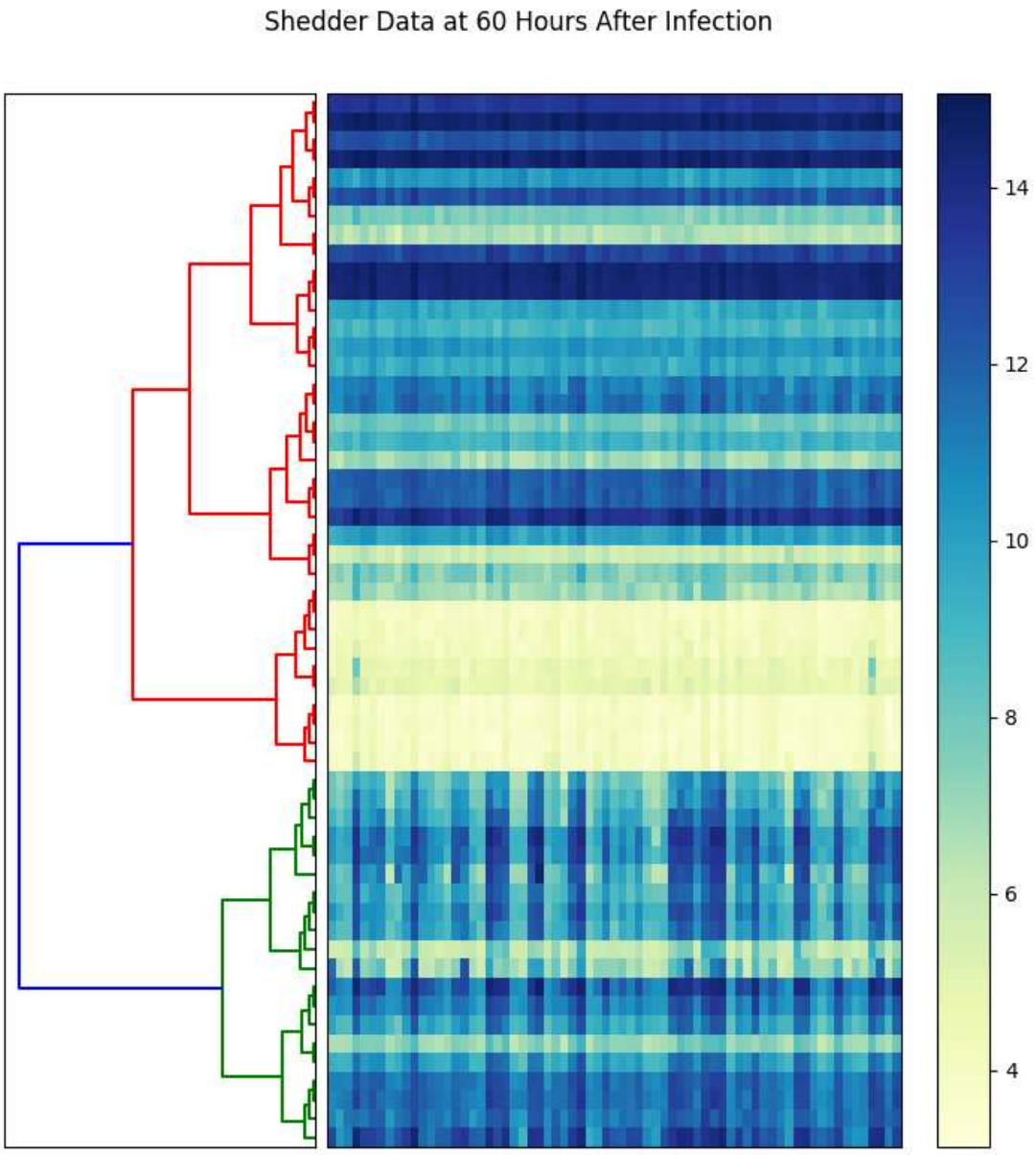


Figure 4.18: Network edges are generated by correlation and clustering is done via ISPA using the Laplacian.

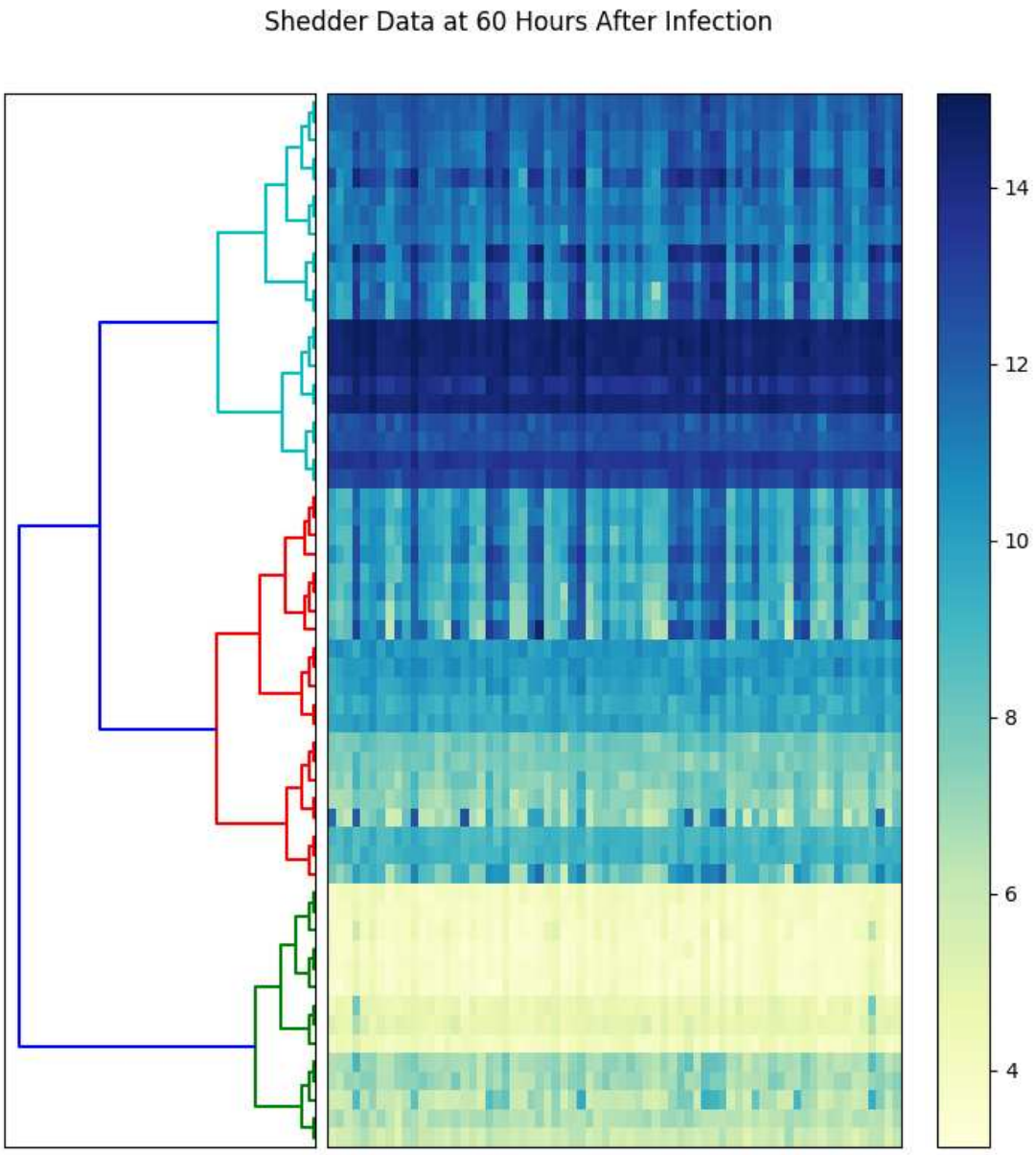


Figure 4.19: Network edges are generated by heat kernel and clustering is done via ISPA using the Laplacian.

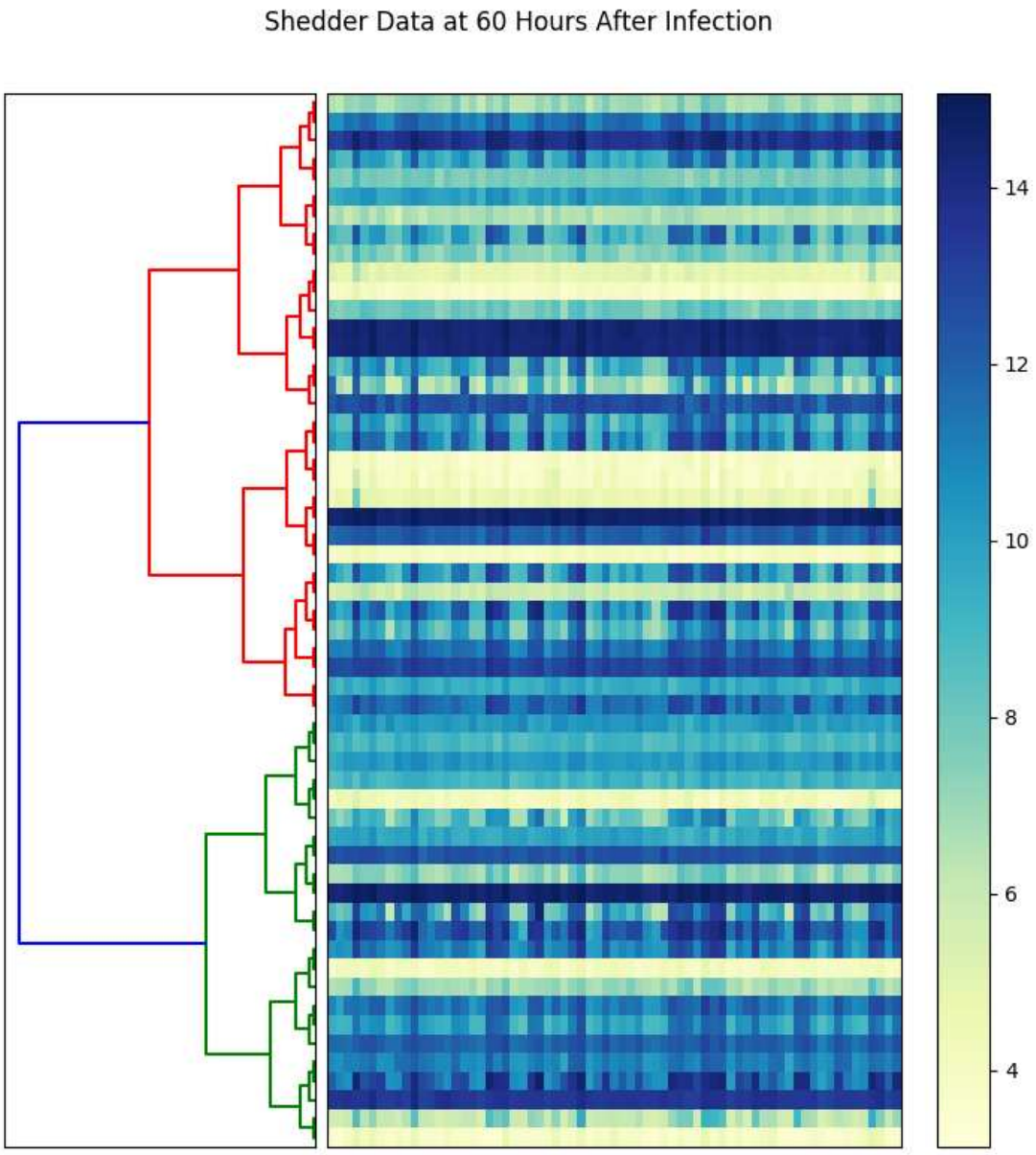


Figure 4.20: Network edges are generated by partial correlation and clustering is done via ISPA using the Laplacian.

At 60 hours after infection it is clear that the method used for defining the edges in the network greatly effects the spectral clustering algorithm. The heat kernel generated network produces the most visually pleasing clustering by grouping together genes with similar expression magnitudes. This is not surprising since heat kernel is used to generate networks for image segmentation which tend to group together pixels with similar colors [5]. Correlation also seems to do some clustering by magnitude, however the larger clusters contain nodes with data that have many different magnitudes. This is because correlation measures the angle between the data points rather than the Euclidean distance. With this in mind, we should be looking for similar left to right patterns in the data nodes in the same cluster. That is to say, nodes in the same cluster have data with the close to the same gene expression levels up to scalar multiples. The dendrogram generated using partial correlation is a bit harder to interpret. However, it is easy to see that the size of the two banks after every cut are about half the size of the larger network, especially true after the first cut.

We also compare these results to a relatively basic WGCNA (weighted gene co-expression analysis). We generate dendrogram following the most simple implementation the process described [6] and leveraging definitions from [18] and [31]. Recall, the ISPA algorithm builds clusters by starting with the whole dataset and recursively partitioning the dataset into subsets until each subset size is one. In contrast, the WGCNA algorithm builds the clusters starting with n different clusters of size 1 and iteratively merges the two closest clusters until we are left with one large cluster containing all the genes. Our definition of close depends on how we define distance between clusters. What follows is one of the many possible implementations of the WGCNA pipeline to our dataset.

The steps of WGCNA were implemented in Python mainly using Scipy packages to generate the dendrogram. We generate a similarity matrix $\mathbf{S}$ using

$$\mathbf{S}_{i,j} := \left|\text{cor}(\mathbf{x}_i, \mathbf{x}_j)^{100}\right|$$

Then we use the topological overlap measure to generate the adjacency matrix $\mathbf{A}$ for the network as

$$\mathbf{A}_{i,j} = \frac{l_{i,j} + \mathbf{S}_{i,j}}{\min\{k_i, k_j\} + 1 - \mathbf{S}_{i,j}}$$

where we define

$$l_{i,j} := \sum_u \mathbf{A}_{i,u}\mathbf{A}_{u,j} \text{ and } k_i := \sum_u \mathbf{A}_{i,u}.$$

Then we use average clustering distance to generate the following dendrogram. The distance between clusters u and v is defined as

$$d(u, v) := \sum_{i,j} \frac{d(u[i], u[j])}{|u| \cdot |v|}.$$

This is exactly the average pairwise distance between points in cluster u and cluster v.

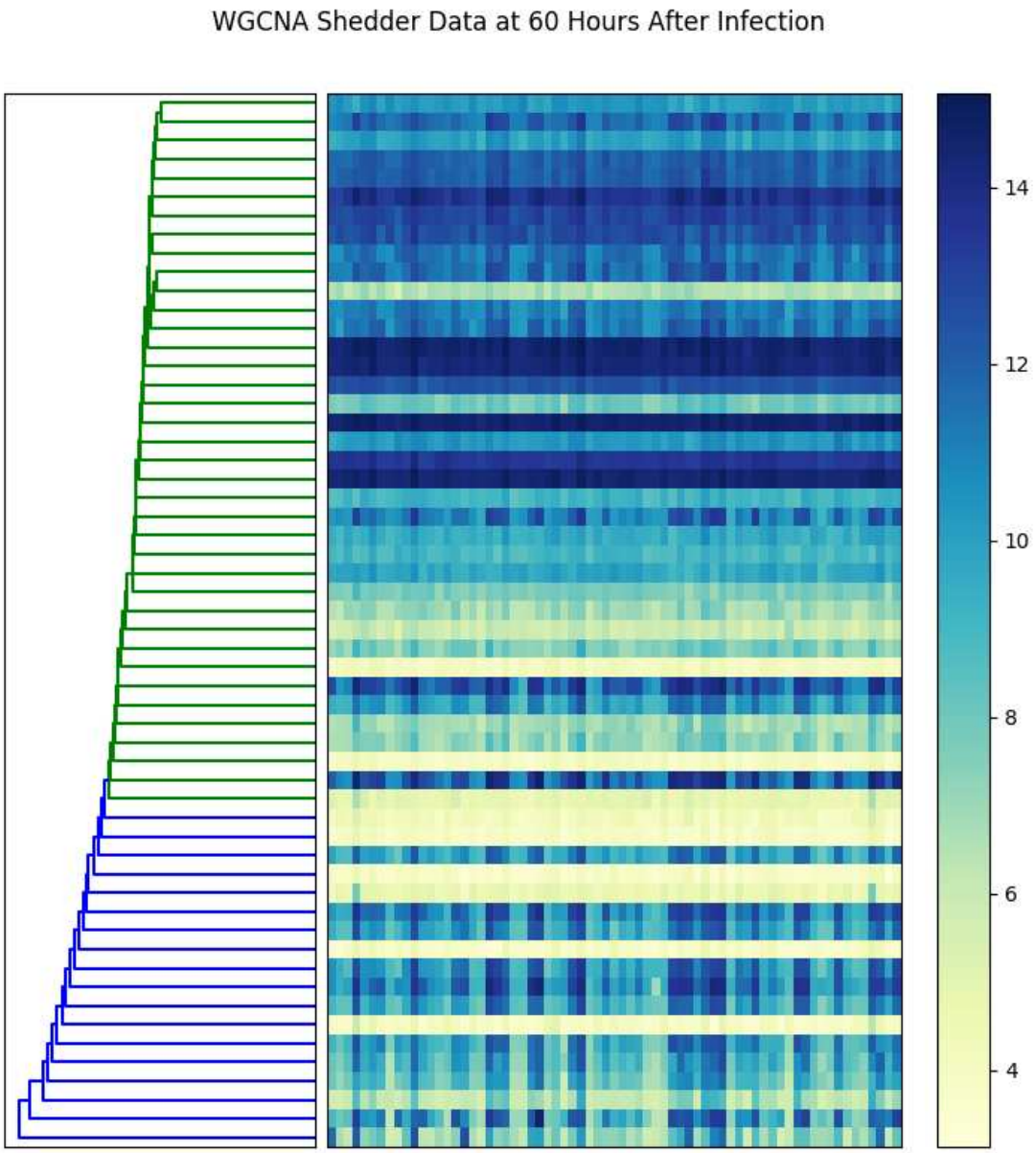


Figure 4.21: Dendrogram generation via a simplified WGCNA algorithm.

This algorithm pulls out many single node clusters. In fact, if we cut this dendrogram at any level, there will be at least one single node cluster. Whereas, the ISPA hierarchical clustering favors separation into even-sized clusters. Also, the distances between clusters using ISPA are relatively larger for the clusters that contain more nodes, whereas the distances between the larger clusters are smaller for the WGCNA algorithm. The distances in ISPA are relatively large for large clusters

because we measure distance proportional to the number of edges cut. For WGCNA, we measure distances using average distance of elements between cluster elements in cluster u and cluster v.

We now search for the perfect module we used the following modified ISP algorithm.

i) Generate a network G using one of correlation, partial correlation, heat kernel or random edge generation

ii) Initialize $M = \min\{w_{i,j} \in A : i \neq j\}$

iii) Apply a threshold with M value to the edges of the original network and proceed to step vii) if the network is disconnected.

iv) Cut the network using the Fiedler vector or the eigenvector associated with the second smallest eigenvalue of the normalized Laplacian.

v) Use SSVM to test each the classification accuracy of each bank. Save the bank, threshold, and accuracy of the higher scoring bank.

vi) Add $0.1(\max\{w_{i,j} \in A\} - \min\{w_{i,j} \in A : i \neq j\})$ and return to iii).

vii) Take the network that produces the highest SSVM value. Call this network G^*

viii) If the accuracy of the SSVM test using the nodes in G^* is lower than the accuracy of the test with G stop and return G, the threshold and the accuracy associated with G. However, if we only did one cut just return G.

ix) Otherwise, let $G = G^*$ and return to ii).

We use only shedder data to generate the network. For our implementation of this algorithm we use the Fiedler vector. We use this algorithm on each time and take our characteristic genes to be the genes that were most common in each G^* when the algorithm terminates. Due to the problem with partial correlation and datasets where $n >> p$, we perform two tests. The first test uses correlation, heat kernel and random edge generation to make networks using data from only

one time point. The second test uses time points that are within 6 hours of each other is the smallest interval which forces $p > n$ so we are able to also use partial correlation to generate the networks. We take the top 15 most common genes in the final G^* over all times greater than 46 hrs for each edge generation method for the first test. For the second test we take the top 15 genes over all time bins with times greater than 36 hours. We test the chosen subset of genes using a two fold SSVM on the gene data for shedders and controls where there are the same number of shedders as controls. A two fold SSVM means we use half the data for training and half for testing. For both halves we ensure there are the same number of shedders as controls.

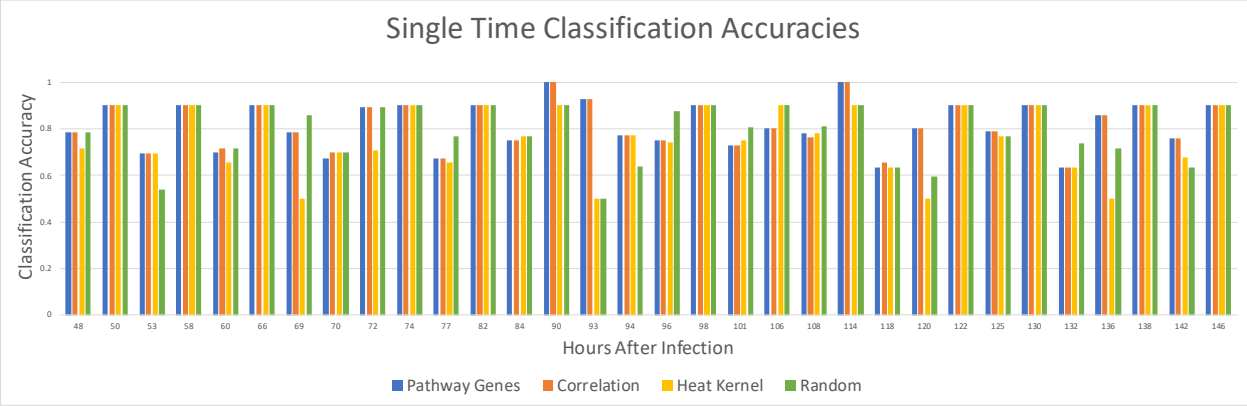


Figure 4.22: The classification accuracies using a 2 fold SSVM for the 15 chosen genes from each edge generation method. These are accuracies for classification on data for each time point.

Most edge generation methods perform equivalently. The top 15 genes from correlation modified ISP algorithm produce accuracies that are on average 0.2 % better than classification accuracies than classification with the entire pathway.

We then choose the 15 most common genes in the final G^* over all times greater than 46 hrs.

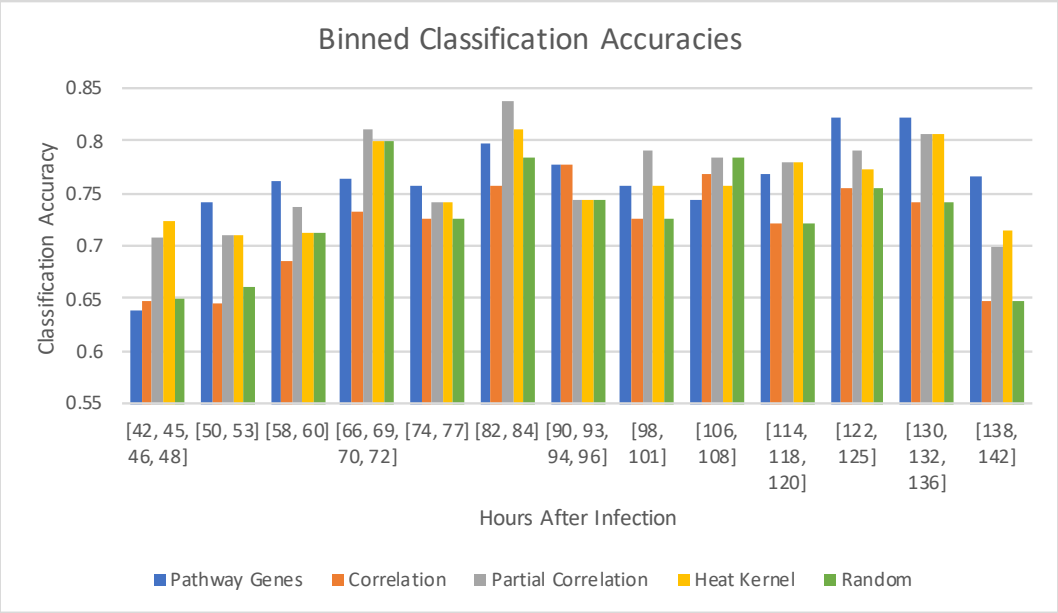


Figure 4.23: The classification accuracies using a 2 fold SSVM for the 15 chosen genes from each edge generation method. These are accuracies for classification on data for each time bin.

Doing this time binning certainly smooths out our classification results and perhaps produces more characteristic feature sets. Partial correlation is on average the best edge generation method. In contrast, correlation does poorly for each time bin. The most consistently good methods for edge generation are be heat kernel and partial correlation It is also worthwhile to note that the random edge generation in general performs better than correlation and almost as well as the other edge generation methods. These results are more realistic than the single time bin results presented earlier because we are working with more data so the data is harder to lineally separate with SSVM.

The best 15 selected genes for time binning within 6 hours and edge generation using partial correlation were the following: IFNA16, IFNA1, OAS3, IFIT3, GBP2, IFNA5, HLA-B, IFITM2, IFITM3, HLA-C, IFI6, HLA-A, EGR1, IRF9, and ADAR. These genes produce a 0.2% better classification rate than classifying with all the genes.

We even compare the two fold SSVM classification accuracies with these 15 genes to classification accuracies using all 12,000+ genes.

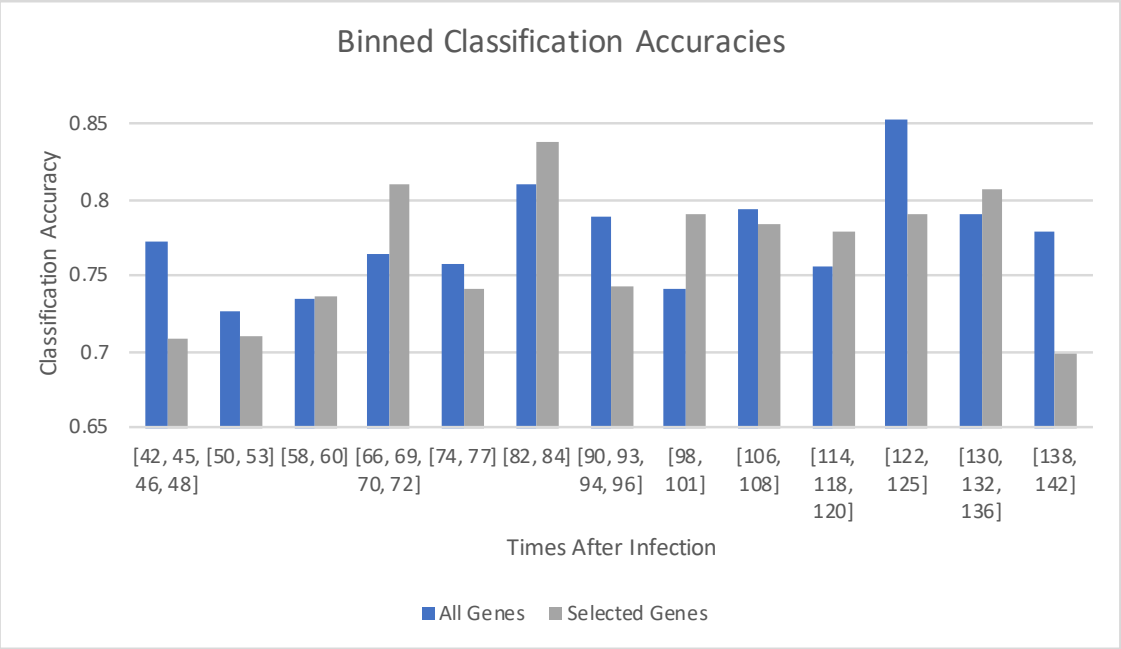


Figure 4.24: The classification accuracies using a 2 fold SSVM for the 15 chosen genes from partial correlation. These are accuracies for classification on data for each 6 hour time bin.

Notice, we produced a feature set of 15 genes which produce higher two fold SSVM classification accuracies than 12,000+ genes.

2) Supra-adjacency Centrality Algorithm

We use the supra-adjacency centrality algorithm with PageRank and largest eigenvector centrality for $k = 1, 2, \ldots, 20$ for networks generated by correlation, partial correlation, heat kernel and random edge generation. We choose to omit partial correlation due to the $n >> p$ issue. The edges between networks at time t and $t + 1$ different times are defined as the mean of the edge weights of all the networks over all times.

For these methods we only use time points between 48hr to 680hr due to prior experimental evidence which suggests that the signal of shedding is most prominent at these times and since a supra-adjacency matrix is build using all times. We then take the characteristic nodes to be the k nodes with the highest two fold SSVM test accuracy for classification on all the given time points.

We perform these tests for $k = 1, 2, \ldots, 20$ with all four edge generation methods along with the two different centrality methods.

With all these tests, the edge generation and centrality methods which produce the highest accuracy are heat kernel and largest eigenvector centrality with $k = 14$ genes. This is determined to be the best method because it produces the highest average SSVM classification rate over single time bins with times at least 48 hours after infection.

The following 14 genes are the top 14 most central genes for the classification that produced a maximum accuracy. IRF8, IFI35, OAS1, IRF3, IFNAR2, IFI6, PTPN1, OASL, IFIT2, ISG15, OAS3, OAS2, TYK2, and IFIT3. 6 of these genes are in the Interferon gamma signaling pathway.

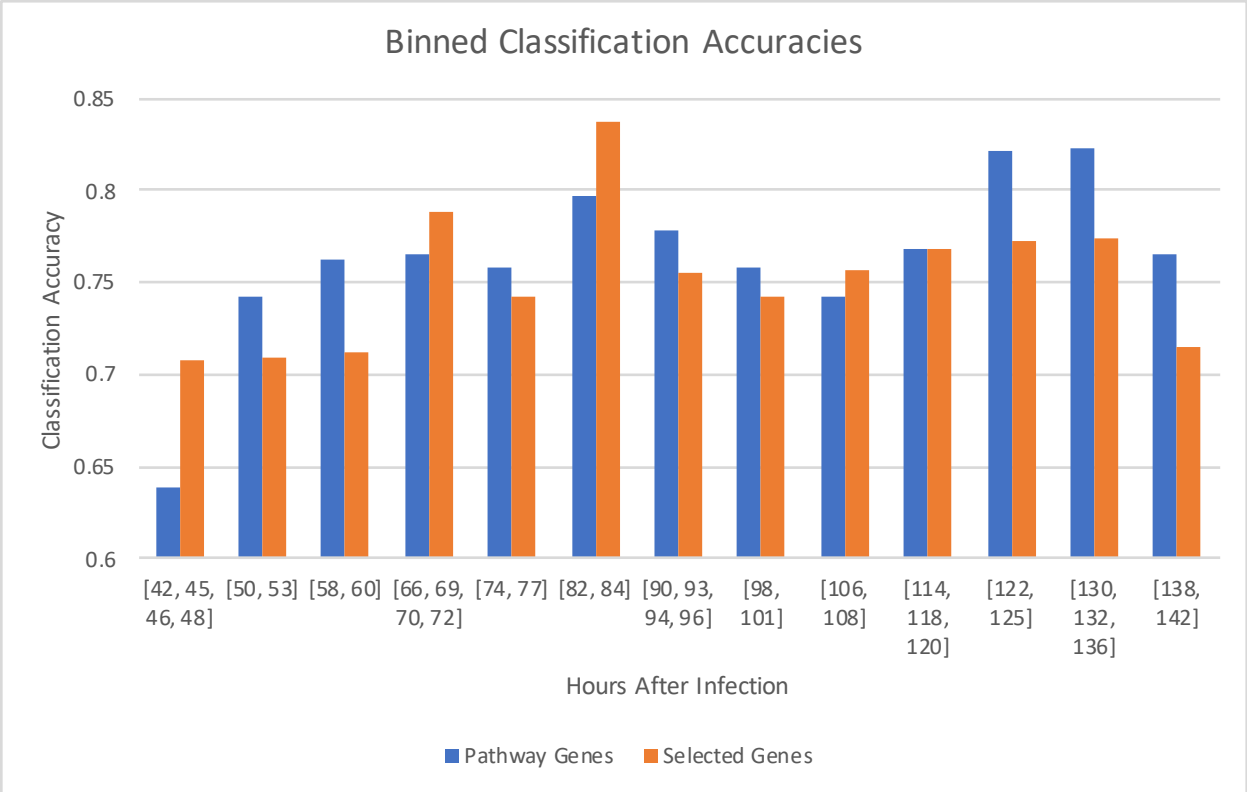


Figure 4.25: The classification results using the 14 most central genes from the supra-adjacency centrality algorithm with heat kernel edge generation and largest eigenvector centrality.

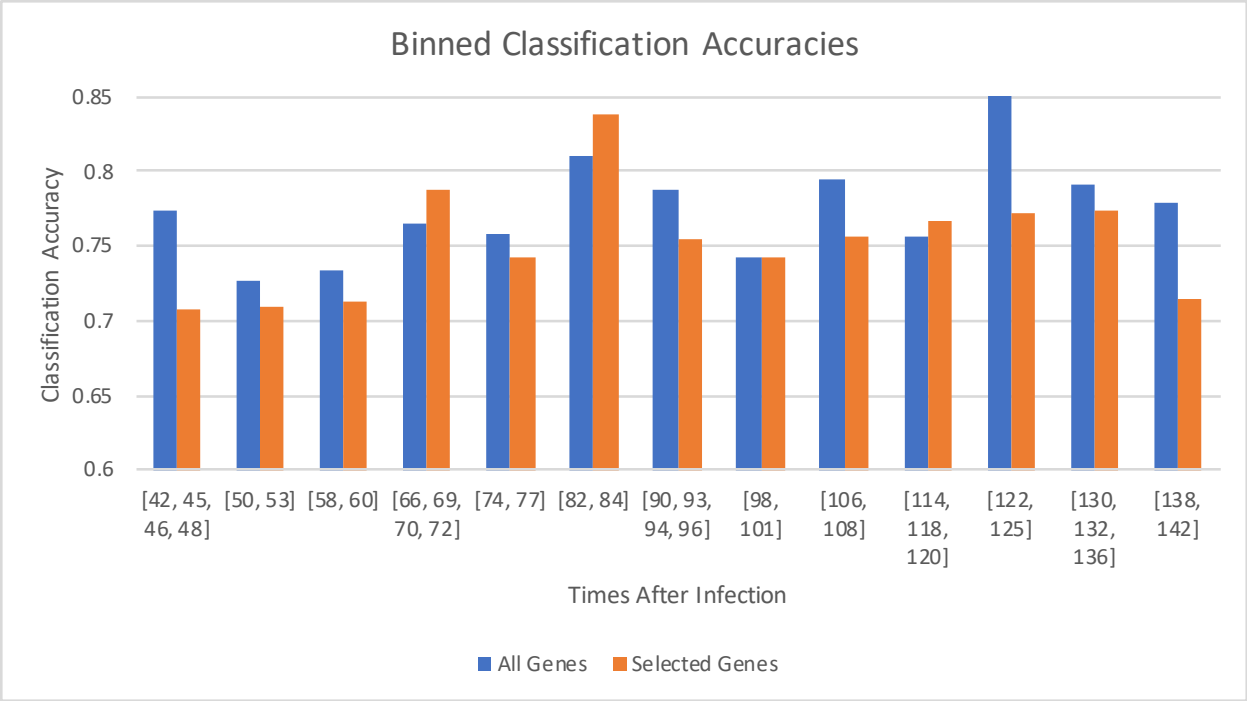


Figure 4.26: The classification results using the 14 most central genes from the supra-adjacency centrality algorithm with heat kernel edge generation and largest eigenvector centrality.

4.2.2 Subjects as Nodes: The Shedder Problem

We use the spectral cut algorithm with the normalized laplacian on a network where subjects are the nodes. For the test without time binning, we generate these networks using one of correlation, heat kernel or random edge generation. We also threshold the edges of the network and choose the threshold which produces the highest classification accuracy.

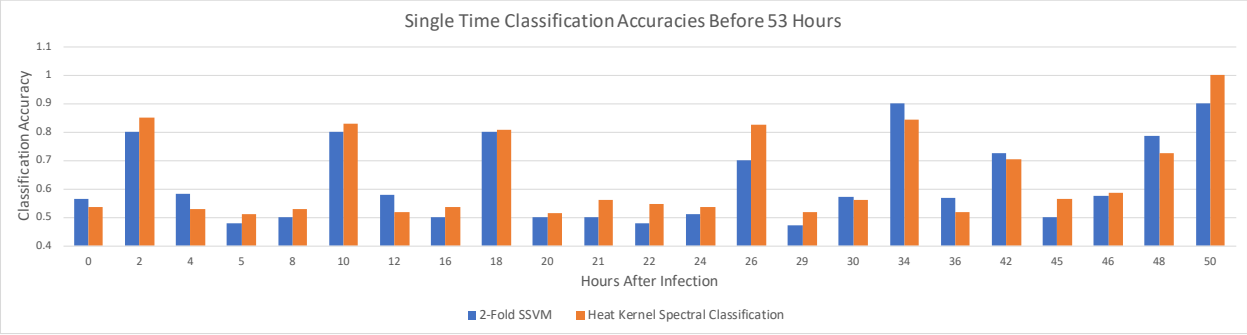


Figure 4.27: The SSVM classification accuracy versus the normalized Laplacian cut unsupervised classification method for heat kernel edge generation for the early times.

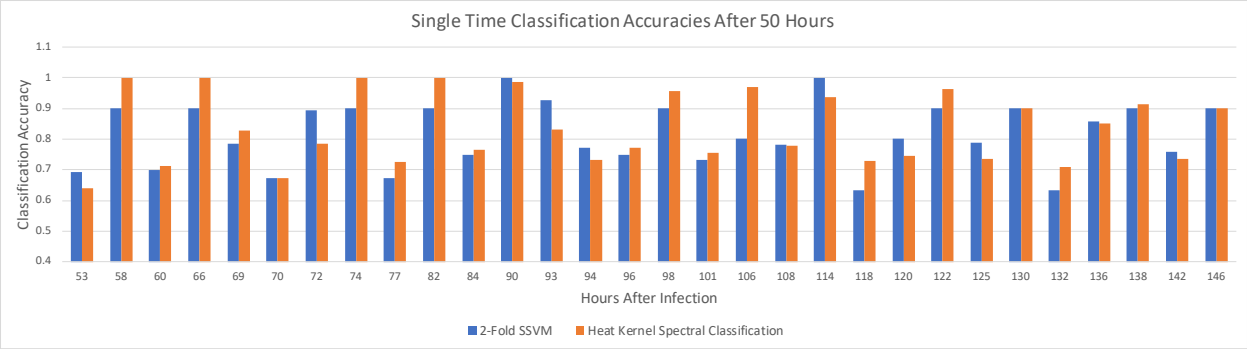


Figure 4.28: The SSVM classification accuracy versus the normalized Laplacian cut unsupervised classification method for heat kernel edge generation for the late times.

Heat kernel edge generation method performs the best, on average almost 2% better than the two fold SSVM.

Since we only have 20 total subjects at some time points, we choose to do classification with 5 hour time bins. This method produces less oscillation in classification accuracies. We consider cuts using the the normalized Laplacian on networks with edges generated by correlation, partial correlation, heat kernel, or random. With this methodology we are not able to produce an unsupervised classifier that classifies almost as well as the two fold SSVM. Again, heat kernel produces this best classification accuracy.

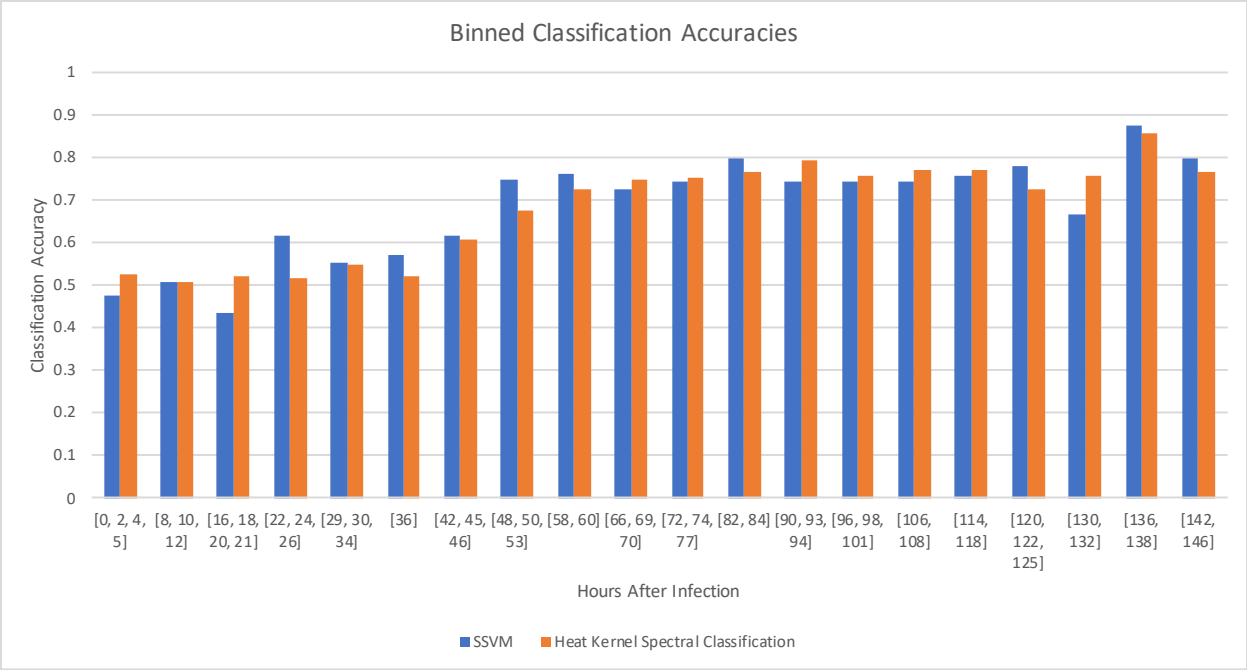


Figure 4.29: The SSVM classification accuracy versus the normalized Laplacian cut unsupervised classification method with no thresholding on adjacency matrix build using heat kernel.

Chapter 5

Conclusion

5.1 Summary

In this paper we began by building a mathematical foundation for our algorithms. We described four different network generation methods. The first algorithm we introduced was a novel version of hierarchical network clustering via iterated spectral partitioning. The other algorithm leveraged measures of centrality on a supra-adjacency matrix. After explaining the mathematics behind our data analysis techniques, we proceeded to provide intuitive examples and visualize our algorithm's results in Euclidean 2-space. Finally, we applied these algorithms to microarray data and found some surprising results. Random graphs seemed to perform equally as well as other network generation methods. In addition, different algorithms suggested different sets of genes for classification. Specifically, there was almost no overlap from the best performing supra-adjacency algorithm and any of the ISPA algorithms. This paper has provided an introduction to spectral network theory algorithms and applies these to a small dataset. In addition, we have managed to identify two sets of genes in the alpha beta interferon pathway which are candidates for biomarkers for shedding with the flu virus. We found that most edge generation methods for either algorithm are relatively equivalent. However, we did notice that partial correlation produced the highest 2 fold SSVM accuracies in conjunction with the modified ISPA algorithms. Whereas, heat kernel coupled with largest eigenvector centrality produced the highest accuracies with the supra-adjacency centrality algorithm. Using human subjects as nodes we were able to barely improve on the two fold SSVM classification accuracies. Since other current research with these data is able to produce much higher accuracies using other supervised learning techniques, perhaps this unsupervised algorithm is not worth investigating further. However, after a long search of the literature we can even be confident in saying this was the first time these spectral network theory algorithms have been used on gene expression data of human subjects sick with the flu.

Highlights from these results include distinguishing two sets of 14-15 genes which produce two fold SSVM classification accuracies at certain times that are at least as high as classification accuracies done with more than 12,000 genes. The different algorithms were used to produce these results. The first is the iterated spectral clustering using the Fiedler vector on a network with edges generated with partial correlation. This procedure selected 15 genes whose two fold SSVM classification accuracy was better than the accuracy the 12,000 genes at the binned times 82 and 84 hours after infection. The second is the supra-adjacency algorithm using largest eigenvector centrality on a network generated with heat kernel. This procedure selected 14 genes whose two fold SSVM classification accuracy was better than the accuracy the 12,000 genes at the two binned time intervals of 74 and 77 hours and 122, 125 hours after infection.

5.2 Future Work

There are a variety of directions for future work with this dataset and these algorithms. On the mathematical front it would be interesting to formulate an expression for the error between the approximate solution to the minimum average cut and minimum normalized cut problems from the graph Laplacian and normalized graph Laplacian respectively. Then we can see what happens to this error as we iterate the network cuts to get some idea of the effectiveness of this algorithm in comparison to Laplacian eigenmaps. One could also attempt to gain a more geometric interpretation of mathematics behind our methodology.

Another step in the testing of these methods would be a sensitivity analysis. That is, checking to see how these methods behave when we add noise to our data.

Also, cleaning up code and providing a well documented public GitHub repository with these algorithms is certainly in the works. Then any researcher could choose to run their data through this analysis pipeline without needing to deeply understand the algorithm design or the mathematics behind it. However, we caution use of these tools without at least some basic understanding of the synthetic examples discussed in this paper.

In terms of the data, a more direct comparison between standard WGCNA and ISPA would be worthwhile. This could entail using each of these clustering in the algorithms then computing the correlation between the eigengenes of each cluster and a given biological trait. This trait does not even necessarily need to be the flu virus.

Another direction for this work could be simply applying these algorithms to the entire GSE-73072 dataset or other large datasets to see if we and discover new pathways. However, this may be computationally expensive depending on the implementation of the algorithms, especially with the eigenvector calculations. One could also just generate the networks and compare them to the known networks structures between genes on the reactome.org pathway browser.

Bibliography

[1] Isabelle Guyon and André Elisseeff. An Introduction to Variable and Feature Selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[2] Yi-Wei Chen and Chih-Jen Lin. Combining SVMs with Various Feature Selection Strategies. In *Feature extraction*, pages 315–324. Springer, 2006.

[3] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature Selection for SVMs. In *Advances in neural information processing systems*, pages 668–674, 2001.

[4] KV Mardia, JT Kent, and JM Bibby. *Multivariate Analysis*. London: Academic Press, 1979.

[5] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural computation*, 15(6):1373–1396, 2003.

[6] Peter Langfelder and Steve Horvath. WGCNA: an R Package for Weighted Correlation Network Analysis. *BMC bioinformatics*, 9(1):559, 2008.

[7] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. ARACNE: An algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context. *BMC bioinformatics*, page S7, 2006.

[8] Antonio Reverter and Eva K. F. Chan. Combining Partial Correlation and an Information Theory Approach to the Reversed Engineering of Gene Co-expression Networks. *Bioinformatics*, 24(21):2491–2497, 2008.

[9] Lun Li, David Alderson, John C Doyle, and Walter Willinger. Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications. *Internet Mathematics*, 2(4):431–523, 2005.

[10] Anna D Broido and Aaron Clauset. Scale-free Networks are Rare. *Nature communications*, 10(1):1017, 2019.

[11] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 07 2014.

[12] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*, 2016.

[13] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[14] Stephen P Borgatti. Centrality and Network Flow. *Social networks*, 27(1):55–71, 2005.

[15] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, 1999.

[16] David Krackhardt. Assessing the Political Landscape: Structure, Cognition, and Power in Organizations. *Administrative science quarterly*, pages 342–369, 1990.

[17] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical Formulation of Multilayer Networks. *Physical Review X*, 3(4):041022, 2013.

[18] Bin Zhang and Steve Horvath. A General Framework for Weighted Gene Co-expression Network Analysis. *Statistical applications in genetics and molecular biology*, 4(1), 2005.

[19] TY Liu, T Burke, LP Park, and CW et al Woods. An Individualized Predictor of Health and Disease Using Paired Reference and Target Samples. *BMC Bioinformatics*, 2016.

[20] Rudolph J Rummel. Understanding Correlation. *Honolulu: Department of Political Science, University of Hawaii*, 1976.

[21] Mechthild Stoer and Frank Wagner. A Simple Min-cut Algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.

[22] Antonio Ortega, Pascal Frossard, Jelena Kovačević, Jose MF Moura, and Pierre Vandergheynst. Graph Signal Processing: Overview, Challenges, and Applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[23] Alex Pothen, Horst D Simon, and Kang-Pu Liou. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM journal on matrix analysis and applications*, 11(3):430–452, 1990.

[24] Miroslav Fiedler. A Property of Eigenvectors of Nonnegative Symmetric Matrices and its Application to Graph Theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

[25] Andrew Duncan. Powers of the Adjacency Matrix and the Walk Matrix. *The Collection*, page 9, 2004.

[26] Phillip Bonacich. Simultaneous Group and Individual Centralities. *Social networks*, 13(2):155–168, 1991.

[27] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The NumPy array: a Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[28] Eric Jones, Travis Oliphant, and Pearu Peterson. {SciPy}: Open Source Scientific Tools for {Python}. 2014.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine Learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[30] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring Network Structure, Dynamics, and Function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[31] Juan A Botía, Jana Vandrovcova, Paola Forabosco, Sebastian Guelfi, Karishma D'Sa, John Hardy, Cathryn M Lewis, Mina Ryten, and Michael E Weale. An Additional K-means Clustering Step Improves the Biological Features of WGCNA Gene Co-expression Networks. *BMC systems biology*, 11(1):47, 2017.

Appendix A

Normalized Minimum Cut Proof

What follows is a more detailed argument that was outlined by the Laplacian Eigenmaps paper [5]. Suppose G is a connected undirected network. For any cut of G into N_1, N_2, let $\mathbf{x}$ be a valuation of the nodes of G where

$$\mathbf{x}_i = \begin{cases} \frac{1}{a} & \text{if } i \in N_1 \\ \frac{-1}{b} & \text{if } i \in N_2 \end{cases}.$$

and $a := \text{Vol}(N_1)$ and $b := \text{Vol}(N_2)$. Using these definitions for the volume of the nodes in each bank, we can rephrase the minimum normalized cut problem as

$$\min \text{Cut}(N_1, N_2) \left(\frac{1}{a} - \frac{1}{b} \right).$$

Now we will prove the following facts:

i) $2\mathbf{x}^T\mathbf{L}\mathbf{x} = \sum_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j}$

ii) $\sum_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j} = 2(1/a + 1/b)^2 \text{Cut}(N_1, N_2)$

iii) $\mathbf{x}^T\mathbf{D}\mathbf{x} = 1/a + 1/b$

The proof of i):

$$\sum_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j} = \sum_i \sum_j \mathbf{x}_i^2 w_{i,j} - 2\sum_i \sum_j \mathbf{x}_i \mathbf{x}_j w_{i,j} + \sum_i \sum_j \mathbf{x}_j^2 w_{i,j}$$

Notice $w_{i,j} = w_{j,i}$ because G is undirected. So

$$\sum_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j} = 2\sum_i \sum_j \mathbf{x}_i^2 w_{i,j} - 2\sum_i \sum_j \mathbf{x}_i \mathbf{x}_j w_{i,j}$$
$$= 2\left(\sum_i \mathbf{x}_i^2 \sum_j w_{i,j} - \sum_i \sum_j \mathbf{x}_i \mathbf{x}_j w_{i,j}\right)$$

Let $d_{i,j}$ be the entry in the ith row and jth column of $\mathbf{D}$. Then

$$\sum_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j} = 2\left(\sum_i \mathbf{x}_i^2 d_{i,i} - \sum_i \sum_j \mathbf{x}_i w_{i,j} \mathbf{x}_j\right)$$
$$= 2\mathbf{x}^T(\mathbf{D} - \mathbf{A})\mathbf{x}$$
$$= 2\mathbf{x}^T\mathbf{L}\mathbf{x}$$

The proof of ii):

$$\sum_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j} = \sum_{i\in N_1, j\in N_1}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j} + \sum_{i\in N_1, j\in N_2}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j}$$
$$+ \sum_{i\in N_2, j\in N_1}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j} + \sum_{i\in N_2, j\in N_2}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j}$$
$$= \sum_{i\in N_1, j\in N_1}\left(\frac{1}{a} - \frac{1}{a}\right)^2 w_{i,j} + \sum_{i\in N_1, j\in N_2}\left(\frac{1}{a} + \frac{1}{b}\right)^2 w_{i,j}$$
$$+ \sum_{i\in N_2, j\in N_1}\left(-\frac{1}{b} - \frac{1}{a}\right)^2 w_{i,j} + \sum_{i\in N_2, j\in N_2}\left(-\frac{1}{b} + \frac{1}{b}\right)^2 w_{i,j}$$

Again we will use the fact that G is undirected. This gives us

$$\begin{aligned}\sum_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2 w_{i,j} &= 2 \sum_{i \in N_1, j \in N_2} \left(\frac{1}{a} + \frac{1}{b}\right)^2 w_{i,j} \\ &= 2\left(\frac{1}{a} + \frac{1}{b}\right)^2 \sum_{i \in N_1, j \in N_2} w_{i,j} \\ &= 2\left(\frac{1}{a} + \frac{1}{b}\right)^2 \text{Cut}(N_1, N_2)\end{aligned}$$

Combining facts i) and ii) we have

$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \left(\frac{1}{a} + \frac{1}{b}\right)^2 \text{Cut}(N_1, N_2)$$

The proof of fact iii) is

$$\begin{aligned}\mathbf{x}^T \mathbf{D} \mathbf{x} &= \sum_i \mathbf{x}_i d_{i,i} \mathbf{x}_i \\ &= \sum_{i \in N_1} \mathbf{x}_i d_{i,i} \mathbf{x}_i + \sum_{i \in N_2} \mathbf{x}_i d_{i,i} \mathbf{x}_i \\ &= \sum_{i \in N_1} \frac{1}{a} d_{i,i} \frac{1}{a} + \sum_{i \in N_2} \left(-\frac{1}{b}\right) d_{i,i} \left(-\frac{1}{b}\right) \\ &= \frac{1}{a^2} \sum_{i \in N_1} d_{i,i} + \frac{1}{b^2} \sum_{i \in N_2} d_{i,i} \\ &= \frac{1}{a^2} a + \frac{1}{b^2} b \\ &= \frac{1}{a} + \frac{1}{b}\end{aligned}$$

Now we can see that our minimum normalized cut problem is equivalent to solving

$$\min_{\mathbf{x} \in \left\{\frac{1}{a}, -\frac{1}{b}\right\}} \frac{\mathbf{x}^T \mathbf{L} \mathbf{x}}{\mathbf{x}^T \mathbf{W} \mathbf{x}}$$

We can relax this problem to be an optimization problem over the set $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^T\mathbf{D}\mathbf{1} = 0\}$. We now will prove $\left\{\mathbf{x} : \mathbf{x}_i \in \left\{\frac{1}{a}, -\frac{1}{b}\right\}\right\} \subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^T\mathbf{D}\mathbf{1} = 0\}$. Take some $\mathbf{x}$ where each $\mathbf{x}_i \in \left\{\frac{1}{a}, -\frac{1}{b}\right\}$. Then

$$\mathbf{x}^T\mathbf{D}\mathbf{1} = \sum_i \mathbf{x}_i d_{i,i} = \sum_{i \in N_1} \frac{1}{a} d_{i,i} + \sum_{i \in N_2} -\frac{1}{b} d_{i,i} = \frac{1}{a}\sum_{i \in N_1} d_{i,i} - \frac{1}{b}\sum_{i \in N_2} d_{i,i} = \frac{a}{a} - \frac{b}{b} = 0$$

Therefore $\mathbf{x} \in \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^T\mathbf{D}\mathbf{1} = 0\}$ so containment has been proven.

Now let $\mathbf{y} = \mathbf{D}^{1/2}\mathbf{x}$. Then the restriction $\mathbf{x}^T\mathbf{D}\mathbf{1} = 0$ is the same as $\mathbf{y}^T\mathbf{D}^{1/2}\mathbf{1} = 0$ because

$$\begin{aligned}\mathbf{y}^T\mathbf{D}^{1/2}\mathbf{1} &= (\mathbf{D}^{1/2}\mathbf{x})^T\mathbf{D}^{1/2}\mathbf{1} \\ &= \mathbf{x}^T{\mathbf{D}^{1/2}}^T\mathbf{D}^{1/2}\mathbf{1}\end{aligned}$$

${\mathbf{D}^{1/2}}^T = \mathbf{D}^{1/2}$ because it is a diagonal matrix. Also $\mathbf{D}^{1/2}\mathbf{D}^{1/2} = \mathbf{D}$. So we have

$$\mathbf{y}^T\mathbf{D}^{1/2}\mathbf{1} = \mathbf{x}^T\mathbf{D}\mathbf{1} = 0$$

and these conditions are indeed equivalent. So setting $\mathbf{x} = \mathbf{D}^{-1/2}\mathbf{y}$ creates the following equivalent optimization problem

$$\min_{\mathbf{y}^T\mathbf{D}^{1/2}\mathbf{1}=0} \frac{\mathbf{y}^T\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{y}}{\mathbf{y}^T\mathbf{y}} = \min_{\mathbf{y}^T\mathbf{D}^{1/2}\mathbf{1}=0} \frac{\mathbf{y}^T\hat{\mathbf{L}}\mathbf{y}}{\mathbf{y}^T\mathbf{y}}$$

If we further relax our constraints to minimizing over all $\mathbf{y} \in \mathbb{R}^n$ then the following minimization problem is approximately equivalent to the minimum normalized cut problem

$$\min_{\mathbf{y}} \frac{\mathbf{y}^T\hat{\mathbf{L}}\mathbf{y}}{\mathbf{y}^T\mathbf{y}}$$

Notice this is the Rayleigh quotient. The eigenvector corresponding to zero is exactly $\mathbf{D}^{1/2}\mathbf{1}$ because

$$
\begin{aligned}
(\mathbf{D}^{1/2}\mathbf{1})^T\hat{\mathbf{L}}(\mathbf{D}^{1/2}\mathbf{1}) &= \mathbf{1}^T\mathbf{D}^{1/2}\mathbf{D}^{-1/2}(D-A)\mathbf{D}^{-1/2}\mathbf{D}^{1/2}\mathbf{1} \\
&= \mathbf{1}^T(D-A)\mathbf{1} \\
&= \mathbf{1}^T D\mathbf{1} - \mathbf{1}^T A\mathbf{1} \\
&= \mathbf{1}^T \begin{bmatrix} d_{1,1} \\ d_{2,2} \\ \vdots \\ d_{n,n} \end{bmatrix} - \mathbf{1}^T \begin{bmatrix} d_{1,1} \\ d_{2,2} \\ \vdots \\ d_{n,n} \end{bmatrix} \\
&= 0
\end{aligned}
$$

The minimum value of the Rayleigh quotient is known to be the second smallest eigenvector of $\hat{\mathbf{L}}$. If we use this vector as a valuation of the nodes of G to define a cut that removes all edges between negative and positive valuated nodes, then we have approximated the solution to the minimum normalized cut problem.

Appendix B

Minimum Cut Proof

Suppose G is a connected undirected network. For any cut of G into N_1, N_2, let $\mathbf{x}$ be a valuation of the nodes of G where

$$\mathbf{x}_i = \begin{cases} 1 & \text{if } i \in N_1 \\ -1 & \text{if } i \in N_2 \end{cases}.$$

Using $i)$ and $ii)$ from the previous section and $a = b = 1$ we have

$$\text{Cut}(N_1, N_2) = \frac{1}{4}\mathbf{x}^T\mathbf{L}\mathbf{x}$$

Also notice

$$\mathbf{x}^T\mathbf{x} = \sum_{i \in N_1} 1 \cdot 1 + \sum_{i \in N_1} -1 \cdot -1 = |N_1| + |N_2| = |N|$$

Therefore we have

$$\text{Cut}(N_1, N_2) = \frac{|N|}{4}\frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \text{ where } \mathbf{x}_i \in \{-1, 1\}.$$

If we relax this restriction on the values of $\mathbf{x}$ to include all real numbers. Then

$$\text{Cut}(N_1, N_2) \approx \frac{|N|}{4}\frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}.$$

For a fixed size network, we have $|N|$ is constant so

$$\arg\min_{\mathbf{x} \in \mathbb{R}} \frac{|N|}{4}\frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathbb{R}} \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$$

This is the Rayleigh quotient. It is known that the non-zero minimum of this is exactly the eigenvector associated with the second smallest eigenvalue of $\mathbf{L}$ (aka the Fiedler vector). So we have shown that the Fiedler vector is an approximate solution to the minimum cut problem. It is im-

portant to note that this is a larger relaxation than the one done in the minimum normalized cut problem $|\{1/a, -1, b\}| \geq 2 = |\{1, -1\}|$.

Appendix C

License

Colorado State University LaTeX Thesis Template

by Elliott Forney – 2017

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication for the benefit of the public at large and to the detriment of our heirs and successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.