BETHE-HESSIAN R OPTIMIZATION

By

Aravind Krishnachandran

Thesis in Major, submitted for
completion of the University Honors Program

June 12, 2020

University of California, Davis

APPROVED

Can Le, Ph.D.

Department of Statistics, University of California, Davis

Table of Contents

Abstract

Detecting communities within a graph or network of data is an important data analysis method that can be applied to many fields of industry. Spectral clustering is a technique used to cluster the data points within a graph into different communities or clusters based on similar factors and connectedness to other data points. One particular spectral clustering algorithm known as the Bethe-Hessian provides estimates of the number of communities and classification of data points to clusters based on the eigenvalues and eigenvectors of the matrix $H_r = (r^2-1)I_n + D - rA$. The efficiency of this algorithm is influenced by the tuning parameter r. I investigate the ideal r value for various sets of tidy and untidy data, and ultimately analyze the properties of the estimated communities as the r value changes. These properties are analyzed by studying the eigenvalues and eigenvectors of the H_r matrix, along with k-means clustering based on these eigenvectors.

1. Introduction

Network Theory is an important area of study as it has applications in several different disciplines, such as sociology, computer science, biology, physics, and more [2]. In network theory, we study the information contained in graphs. A graph *G(V, E)* consists of *V* nodes and *E* edges. Nodes are the data points, and edges are the linkages between these data points. Graphs can be directed or undirected, and weighted or unweighted. In the case of directed graphs, there are directions associated with edges linking two nodes, while there is no edge direction in unweighted graphs. In the case of a weighted graph, the edges connecting two nodes may have a weight associated with it, while the edges of an unweighted graph have a weight of 1 if an edge exists between two nodes. An example of an unweighted and undirected social network graph is given in Figure 1. One can think of the nodes as people, and the edges indicate that the people are friends. The blue nodes indicate one community of friends, while the red nodes indicate another community.

A community (also referred to as a graph cluster) consists of nodes with similar characteristics. The goal of community detection is to identify the potential underlying communities within the framework of a graph and partition the nodes into these communities [3]. Detecting communities is an important task as it can help identify items which may be closely related in some way. This can have significant applications in a given field.

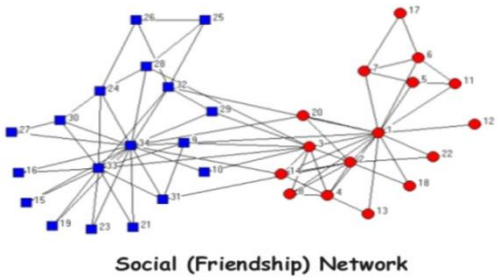

Figure 1: A social network graph.

For example, clusters determined in a network of customers and their purchasing habits can help provide customers with recommendations on products they may want to buy [8]. However, detecting communities can be difficult, especially because in most real-world scenarios, one does not know how many underlying communities there truly are within a network. Algorithms may also struggle at times to determine the correct assignment of nodes to clusters. Fine-tuning these algorithms may be necessary to detect communities more efficiently. Therefore, community detection can be a challenging problem of interest.

2. Literature Review and Background

Communities can be split into two types. Assortative communities are communities where nodes of the same community are densely connected, meaning that there are more edges connecting these nodes than edges connecting these nodes to nodes elsewhere [3]. These types of communities are more commonly seen. Disassortative communities, on the other hand, contain nodes that have higher external connectivity.

A common and popular method to detect communities in graph theory is spectral clustering. Spectral clustering clusters nodes based on a similarity matrix [9]. It involves looking at the eigenvectors of this matrix, and using the elements of these eigenvectors to cluster the graph into communities. Rather than utilizing the points directly to cluster, the elements of the eigenvectors are utilized since the eigenvectors aid in making the cluster properties of the graph data more apparent [8]. The number of eigenvectors used for clustering depends on the eigenvalue distribution. One can apply the unsupervised k-means clustering algorithm on the eigenvectors of the similarity matrix to obtain the community structure. K-means clustering is unsupervised as there is no target response to be predicted. K-means takes a value *k* from the user to initialize as the number of centroids. It calculates the distance between all points to the centroids and assigns the points to the cluster associated with the centroid. Iterative calculations are performed until the optimal centroid positions are obtained [7].

The stochastic block model is a random graph model which can be used to generate a graph based on user-specified parameters. One can specify the number of nodes in each community (or block). One can also specify the probability that the nodes of one community connect with the nodes of another community through a probability matrix. Every edge is drawn independently [4]. The generated graph labels can be used to compare with the results of a

clustering algorithm to measure how well the clustering algorithm performed in partitioning nodes into communities.

Previous literature [1,4,5] has shown that a matrix formula known as the Bethe-Hessian is a spectral clustering algorithm that can be used to identify communities. The formula is given by

$$H_r = (r^2 - 1)I_n + D - rA$$

The resulting matrix is a real-valued, symmetric matrix. I_n indicates the identity matrix, consisting of ones on the diagonal and zeros on all other indices. D is a diagonal matrix consisting of elements on the diagonal denoting the degree of each node. A degree is the number of edges associated with a node. A represents the adjacency matrix of a graph. For unweighted and undirected graphs, an adjacency matrix is a symmetric matrix which has ones where two nodes are connected by an edge, and zeros otherwise. Lastly, r is a tuning parameter. A tuning parameter is a parameter in a machine learning algorithm used to control the algorithm's behavior. One can fine-tune a tuning parameter to optimize the algorithm for the situation it is being used in.

Studying the eigenvalues and their corresponding eigenvectors resulting from H_r are important to understanding the community structure outputted by the algorithm. The authors of [1] indicate that one can look at the number of negative eigenvalues of H_r to determine the number of underlying communities within a graph. The authors indicate that $|r| > 1$, and that the ideal value for r should be $\sqrt{c}$ (where c indicates the average degree) in the case of a stochastic block model. In the general case, they indicate that it should be $\sqrt{\rho(B)}$, where $\rho(B)$ is the spectral radius of the non-backtracking operator. The non-backtracking operator is a method of clustering graphs into communities, which was shown to be outperformed by H_r in [1]. $\rho(B)$ can

be estimated by $\sum_i d_i^2 \,/\, \sum_i d_i$, where d_i is the degree of node i [5]. Negative r values can be used to identify disassortative communities, while positive r values can be used to identify assortative ones [1].

To cluster in the case of two communities, one can use the signs of the eigenvector associated with the second smallest eigenvalue (positive values indicate the node belongs to one community while negative values indicate the node belongs to the other community) [1]. In the general case, one can apply the k-means clustering algorithm to the eigenvectors associated with the k negative eigenvalues of H_r as one does in spectral clustering.

The Bethe-Hessian was noted in [2] to overcome sparse networks (networks where the degrees of the nodes are much smaller than the number of the nodes). The authors of [2] noted that the choice of r in [1] can be suboptimal in cases where graphs are generated by the degree-corrected stochastic block model, where the graphs are sparse and heterogeneous. Many real graphs are heterogeneous, meaning that the distribution of degrees among nodes is far from identical.

Two possible ways to measure the performance of a clustering algorithm's partition of nodes are overlap and modularity. The authors of [1] provide a formula for calculating the overlap between the estimated labels and the ground-truth labels. This is given by

$$\left(\frac{1}{N}\sum_u \delta_{t_u,e_u} - \frac{1}{q}\right) \Big/ \left(1 - \frac{1}{q}\right)$$

with t_u representing the true label of node u, e_u representing the estimated label, and maximization over all $q!$ possible permutations of the groups taking place. Modularity is a measure of the quality of a partition of nodes. It can be calculated by

$$\frac{1}{2|\varepsilon|}\sum_{i,j=1}^{n}\left(A_{ij}-\frac{d_i d_j}{2|\varepsilon|}\right)\delta\left(l_i,l_j\right)$$

[5, 8]. A higher modularity means that more edges fall into a cluster than would be expected by chance.

3. The Study

The purpose of this research study is to get a better understanding of the tuning parameter *r* of the Bethe-Hessian machine learning algorithm. Specifically, the analysis consists of determining ideal values of *r* for various real graphs. I also explore how varying *r* impacts the communities that are returned from the Bethe-Hessian algorithm, as well as what the characteristics and properties of these generated communities are.

To conduct the research, I utilize the programming languages R and Python to run various simulations and observe patterns associated with different values of *r* for different sets of data (both real graphs and randomly generated graphs). Specifically I utilize the packages "igraph" and "networkx" to perform several operations with graphs, including calculating degrees, constructing adjacency matrices, and plotting overlap/modularity for varying *r* values. I focus on undirected and unweighted graphs in the study.

I analyze the real networks provided on Mark Newman's Network Data webpage [6], which has links to GML files containing real network datasets. I utilize the stochastic block model to perform simulations on modeled graphs and observe general patterns associated with different *r* values.

4. Research Findings

4.1 Real Network Data Analysis

4.1.1 Dolphins Network

The dolphins network consists of 2 true communities, with one community containing 42 nodes and the other containing 20 nodes. If we utilize $\sqrt{c}$ (2.265) and $\sqrt{\rho(B)}$ (2.609) as the values of r in the Bethe-Hessian matrix formula, we can see that in both cases the two smallest eigenvalues are indeed negative and are separated from the rest of the bulk. This indicates that there are two communities, matching up with the true number of communities.

After calculating the overlap between the true labels and the estimated labels, the maximum overlap is attained for r vales ranging from 1.0 to 2.2, with an overlap of 0.968. The modularity also attains its maximum in this range, with the modularity being 0.379. This means that $\sqrt{c}$ is an ideal r value, while $\sqrt{\rho(B)}$ is not. Node number 39 is the only node misclustered (clustered into the smaller community when it was originally in the larger community). From the graph we can see that the node is in the area where the two different communities appear to separate. This may indicate that nodes on the fringes of communities may be susceptible to being clustered into the wrong community.

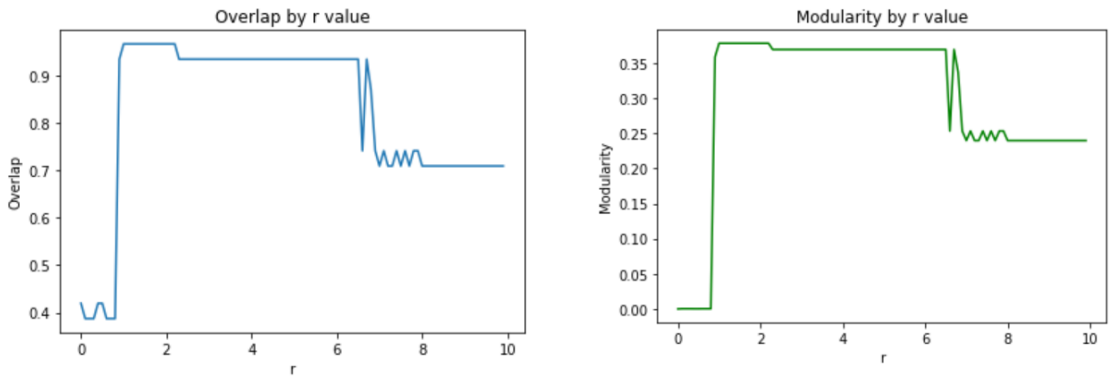

Figure 2: Overlap and modularity by *r* value for Dolphins Network.

4.1.2 Zachary's Karate Club Network

The karate club network consists of 2 true communities, with one community containing 16 nodes and the other containing 18 nodes. If we utilize $\sqrt{c}$ (2.142) and $\sqrt{\rho(B)}$ (2.787) as the values of r in H_r, we can see that only in the former case the two smallest eigenvalues are negative. In the case of $r = \sqrt{\rho(B)}$, we see that the second smallest eigenvalue is actually positive. I found that choosing a very large value (such as 100) helped to distinguish that there are two underlying

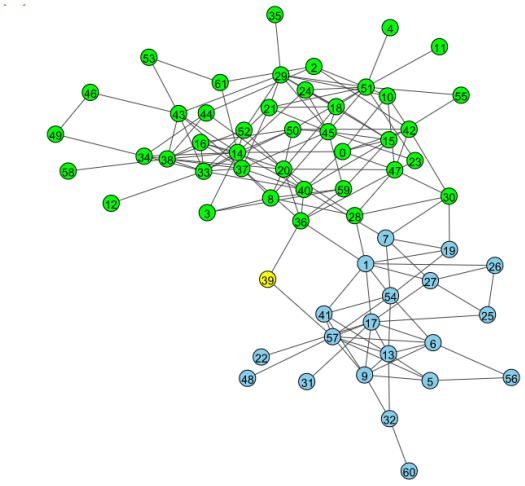

Figure 3: The Dolphins Network. Node number 39 is the only node to be clustered into the other community (colored in yellow to indicate that it is misclustered).

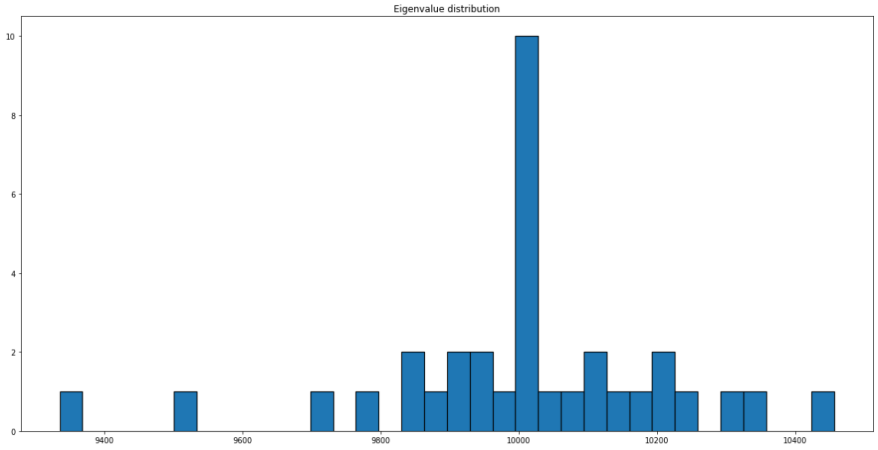

Figure 4: Eigenvalue histogram
for r = 100.

communities. Instead of looking for the negative eigenvalues, I looked at the distance between isolated eigenvalues and the bulk. One can see from Figure 4 that the greatest distance actually occurs between the second and third smallest eigenvalues, possibly indicating that distance from the bulk may sometimes be a better indicator of the number of underlying communities rather than eigenvalues being negative.

Calculating the overlap between the true labels and the estimated labels, we see that maximum overlap is attained for r vales greater than or equal to 1.5, with an overlap of 1, meaning that there are no misclustered nodes. The modularity also attains its maximum in this range, with the modularity being 0.371. This indicates that both $\sqrt{c}$ and $\sqrt{\rho(B)}$ can be optimal choices for r here.

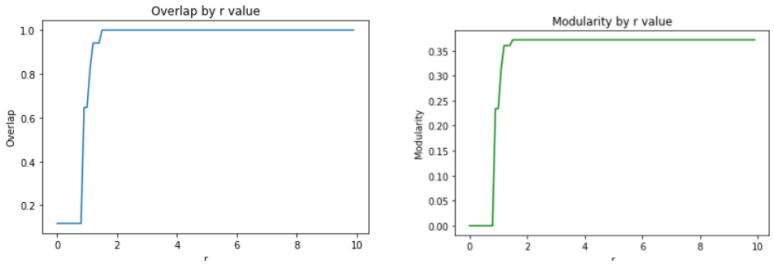

Figure 5: Overlap and modularity by *r* value for Karate Network.

4.1.3 Political Books Network

The political books network consists of 3 true communities, with the smallest community containing 13 nodes, and the two larger communities containing 49 nodes and 43 nodes. For this graph, I noticed that $r = \sqrt{\rho(B)}$ indicated that there are 3 negative eigenvalues, matching up with the 3 underlying communities. The optimal *r* value that maximizes overlap is 0.8, with an overlap of 0.829. However, in Figure 7, we can see that only two nodes are clustered into the smallest group. For $r > 1$, values in the range of 1.1 to 1.4 maximize overlap (overlap of 0.771). These *r* values lead to more nodes in the smallest cluster, potentially giving credence to the claim made by [1] that $|r| > 1$. The ideal *r* value in terms of modularity is in the range between 2.1 to 2.4 (modularity of 0.518). Neither $\sqrt{c}$ (2.898) nor $\sqrt{\rho(B)}$ (3.455) are in the maximal overlap or maximal modularity ranges. Unlike the previous two graphs, the modularity curve does not match up with the overlap curve, and modularity peaks after overlap peaks. This could be due to the presence of three communities as opposed to two.

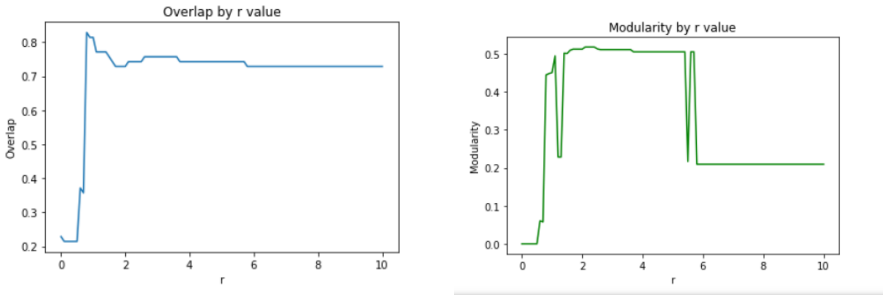

Figure 6: Overlap and Modularity by *r* value for Political Books Graph.

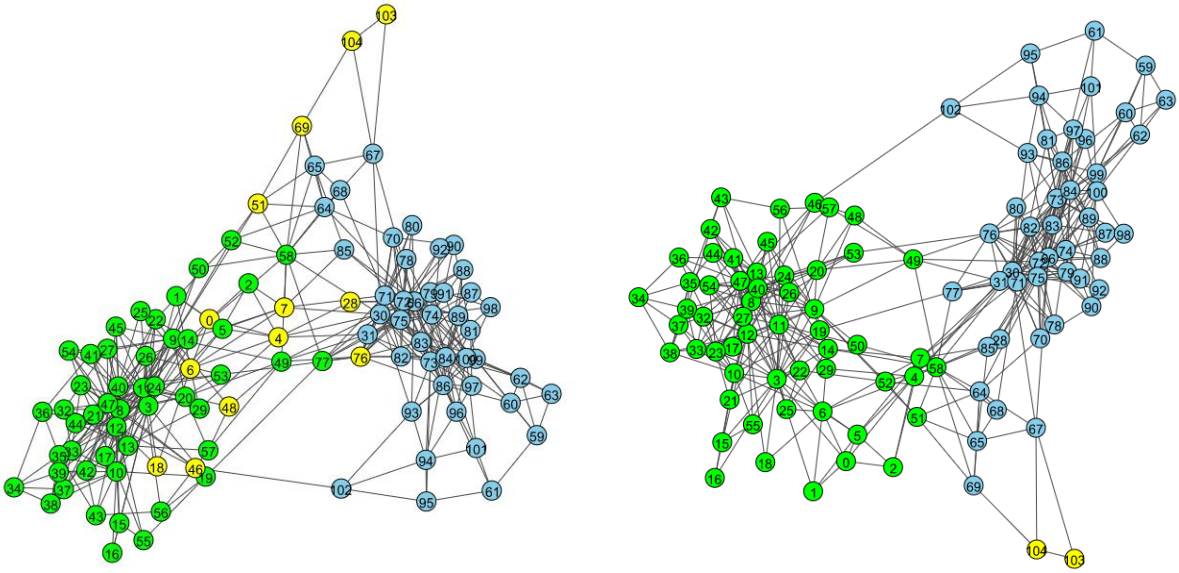

Figure 7: On the left is the Political Books graph with the true labels colored. On the right is the Political Books graph with colored nodes based on estimated labels from $r = 0.8$. Most of the nodes from the smallest group have been clustered into the two larger communities.

4.1.4 Word Adjacencies Network

The word adjacencies network consists of 2 true communities, with one community containing 58 nodes and the other containing 54 nodes. This network is one of the uncommon disassortative communities.

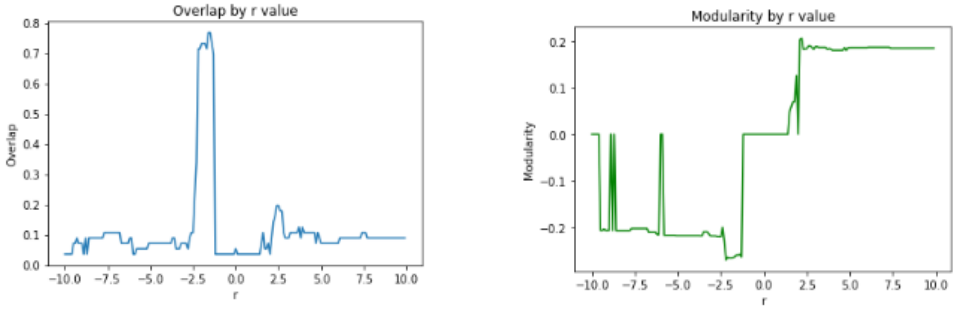

Figure 8: Overlap and Modularity by r value for Word Adjacencies Network.

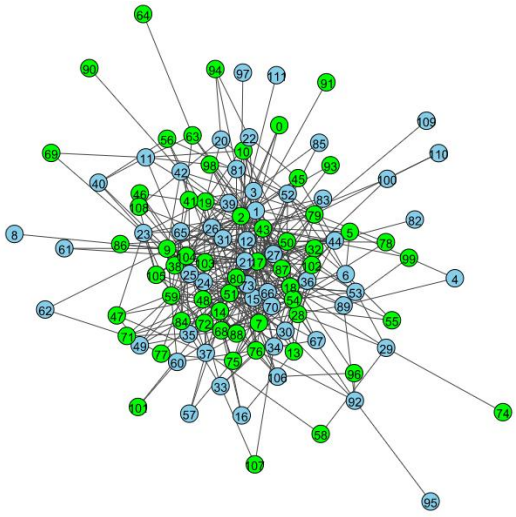

Figure 9: The Word Adjacencies network with the true node community labels indicated by either green or blue. The disassortativity results in a network with several edges connecting nodes of differing communities.

I was able to determine this by calculating the overlap for various *r* values. I noticed that the overlap peaks for *r* values in the range from -1.6 to -1.5, with an overlap of 0.768. This is higher than any positive *r* value. Unexpectedly, the modularity is positive for positive *r* values. It is lowest at *r* = -2.1, with a modularity of -0.267, while it is highest at *r* = 2.2, with a modularity of 0.206. The overlap appears to peak around where modularity is most negative. With an overlap of 0.768, 13 nodes end up being misclustered when *r* is set to -1.5. Neither $-\sqrt{c}$ (-2.755) nor $-\sqrt{\rho(B)}$ (-3.711) are in the maximal overlap or modularity ranges, meaning they are not ideal choices for *r*.

4.1.5 Football Network

The football network consists of 12 true communities, with a total of 115 nodes, meaning that each community is not very large. Using the negative eigenvalue approach here is not effective, since there are only 10 negative eigenvalues but there are 12 communities for both *r* =

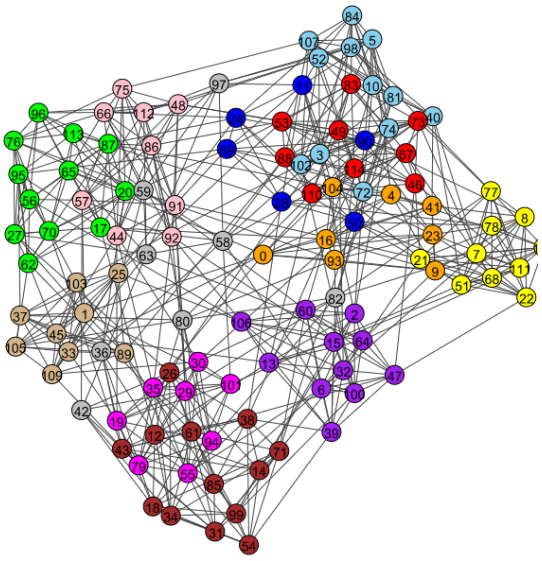

Figure 10: The Football Network with the true node community labels indicated by various colors.

$\sqrt{c}$ (3.265) and $r = \sqrt{\rho(B)}$ (3.276). Therefore, I checked to see which collection of eigenvectors helped result in the highest possible overlap and highest possible modularity. I found that the eigenvectors that result in the highest overlap and modularity are the 9th and 11th smallest, not the 12th smallest. Looking at Figure 11, we can see that the 11 smallest eigenvalues appear to be isolated (there is a larger distance from the 11th to the 12th). Therefore, I utilized the 11 smallest eigenvectors to cluster, and reached an overlap of 0.991 by $r = 1.1$.

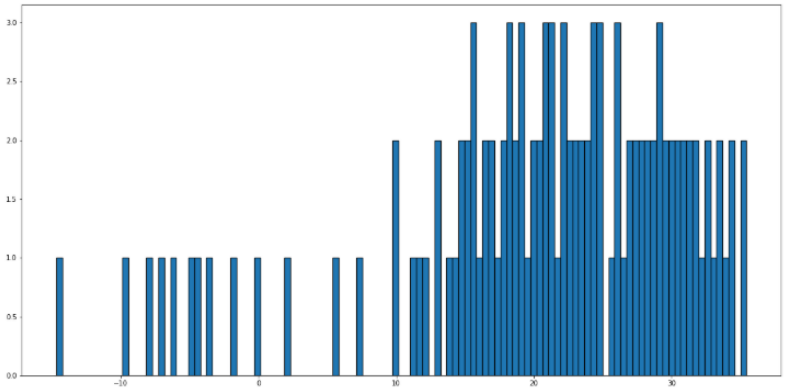

Figure 11: Histogram of eigenvalues for $r = \sqrt{c}$ for Football Network.

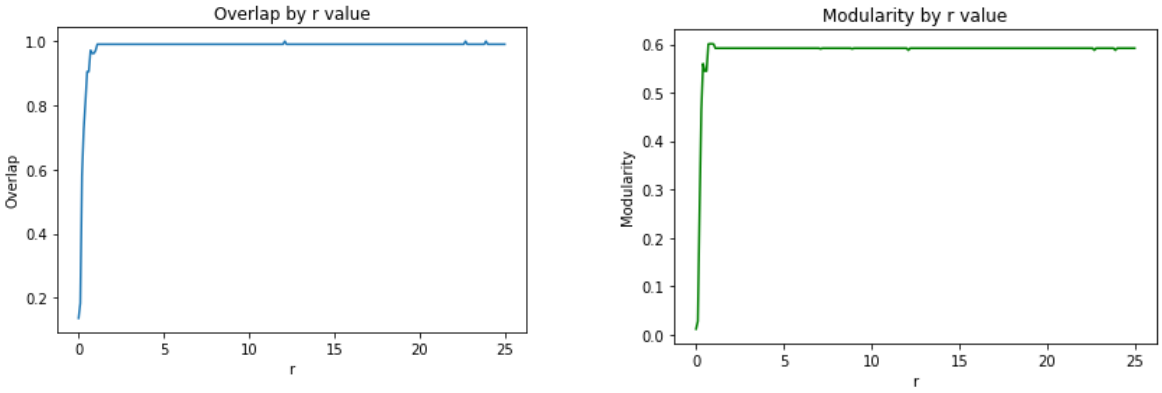

Figure 12: Overlap and modularity by *r* value for Football Network (11 eigenvectors). Overlap flattens out but reaches a higher peak for larger r values, while modularity peaks, dips, and stays flat for larger r values.

However, when I extended the range of *r* values, I noticed that I was able to attain an overlap of 1 for a few choices of *r*. The earliest such case is $r = 12.1$. This may indicate that for cases where there are several communities, an unusually large *r* value can be slightly more efficient for partitioning nodes into communities. The maximum modularity was attained for *r* values in the range from 0.8 to 0.9 (0.6010). When I used 12 eigenvectors to cluster, the maximum overlap I was able to achieve was 0.972 for any *r* value greater than or equal to 0.6, and a slightly lower maximum modularity of 0.6005 for the same range of *r*.

4.2 Stochastic Block Model Simulations

I used the stochastic block model to generate graphs with varying block sizes and varying intercommunity edge probabilities along with varying intracommunity edge probabilities to observe how the value r in H_r affected the quality of the clustering in each of these situations. I mostly focused on uneven block sizes, since large real networks typically do not have the same number of nodes in each community.

One common pattern that was noticed among almost all generated graphs was the spikes in modularity and overlap that take place after a certain r value has passed. This r value is usually between 0 and 1. For r values very close to zero, almost all the nodes are grouped into one large cluster (all represented by one color), barring a few which are initialized as the centroids for other clusters. Gradually increasing r, once a certain r value is passed, one can see several nodes from the large group be clustered into one of the other sparse groups. As this cluster gains many new members, one can see a large increase in the overlap, along with the modularity.

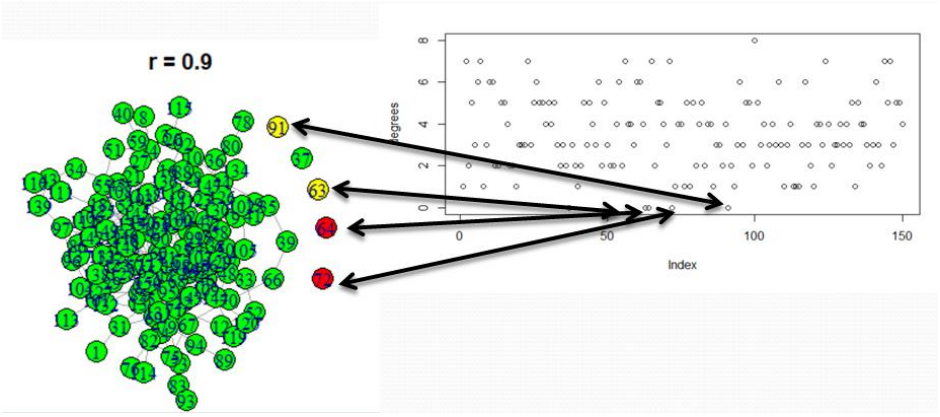

Figure 13: On the left is a plot of a graph generated from the stochastic block model, clustered according to the three smallest eigenvectors of H_r for $r = 0.9$ (3 true clusters). On the right is a scatter plot depicting the degrees of each node. Nodes with the smallest degrees are the first ones to be initialized as separate clusters from the green bulk of nodes.

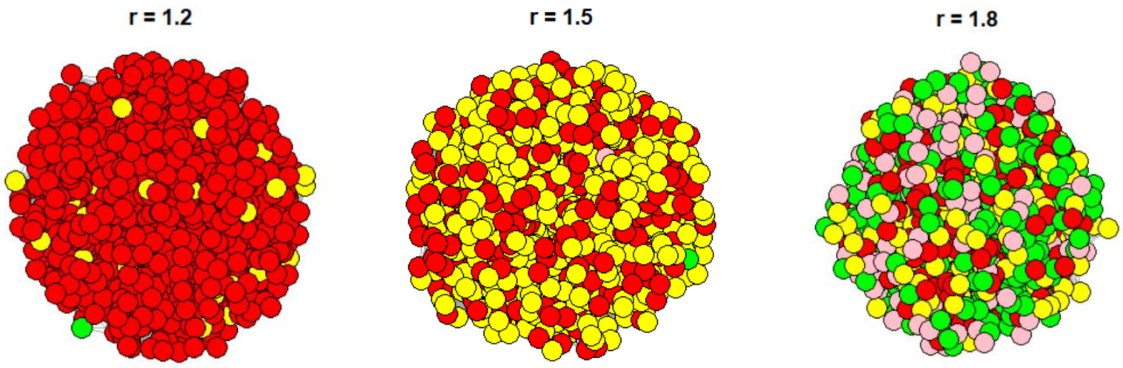

Figure 14: The formation of clusters is rapid when one increases the value of r for values of r approximately less than 2. At r = 1.2 we see most nodes are clustered into one community, but by r = 1.8, we see a prevalence of nodes labeled with all the different community colors. This graph is generated from a stochastic block model with within edge probability of 0.04 and between edge probability of 0.03. The blocks are of uneven sizes (block sizes = 200, 450, 700, 950).

In the case of 3 clusters, as the third cluster gains more members, one can also see another increase in the overlap and modularity. However, this time the spike is a smaller spike compared to the formation of the second cluster. This behavior was also seen in the real networks studied earlier.

In the cases where within community probabilities are much greater than between community probabilities (e.g. within community probability of 0.25 and between community probability of 0.05), $\sqrt{c}$ and $\sqrt{\rho(B)}$ work well in determining the number of communities through the number of negative eigenvalues of H_r. This is because communities are more well-defined in these cases. It is also possible to achieve an overlap equal to 1 (full efficiency) or an overlap very close to 1, and modularity above 0.25, regardless of block size. For these cases, large r values tend to either lead to a decline or a plateau in both overlap and modularity. The nodes that are misclustered tend to be the nodes which originate from the smallest community, while the nodes from the largest community are, if not perfectly clustered, representative of the

true node labels from that community. Also in these cases, larger block sizes tend to have nodes with higher degree values relative to nodes of smaller blocks. However, this is swapped in the case of disassortative communities, meaning that nodes from the smaller blocks will tend to have the most number of edges associated with them. The stochastic block model simulations indeed show that negative *r* values fare better than positive ones for disassortative communities.

When all the probabilities in the stochastic block model edge probability matrix are small, the network becomes a sparse network, with the average degree being much smaller than the number of nodes. This means that $\sqrt{c}$ and $\sqrt{\rho(B)}$ will be low numbers, indicating that a possible ideal *r* value will tend to be small as well. In the interesting case where all probabilities are low, and within and between edge probabilities are very close, we can see that there are spikes in overlap before *r* is even positive (assortative case). There is a range around 0 where *r* tends to produce higher overlap values, and past this range the overlap performance dips, as can be seen in Figure 16. It is also interesting to note that modularity is slightly negative in the range overlap peaks. Overlap and modularity are quite low in these cases, which is to be expected since the communities are much less defined. Overlap and modularity continue to decline with larger *r* values.

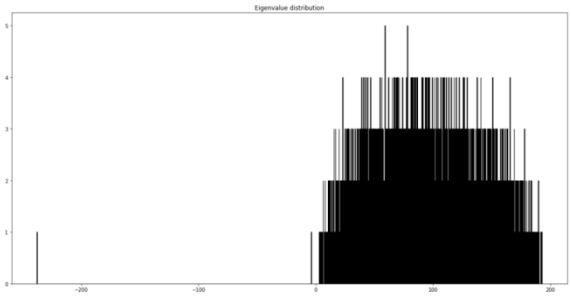

Figure 15: Eigenvalue distribution for $r = \sqrt{\rho(B)}$ for graph with 4 true communities (uneven block sizes, within probabolitiy is 0.05, without probability is 0.03). Only two eigenvalues are negative, which is supposed to indicate that there are two true communities, while there are actually 4.

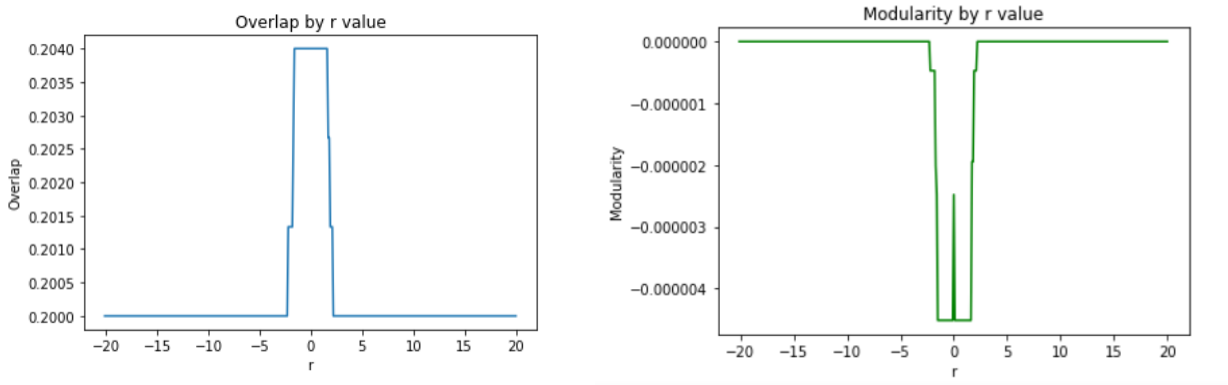

Figure 16: Overlap and modularity by *r* value for a stochastic block model graph with 4 communities, uneven block sizes, within probability 0f 0.1, and without probability of 0.09. When the probabilities are so close, the ideal *r* value hovers around zero. Negative *r* values work here even though the communities are assortative.

I also studied cases where I assumed that I did not know the true number of communities and viewed how nodes were clustered when using an incorrect number of centroids for k-means clustering. I utilized the eigenvectors associated with the *k* smallest eigenvalues (where *k* represents the assumed number of communities) and looked at sparse cases. For example, in the case of 4 true communities/blocks, I assumed that there are only 3 communities. When I did this, I typically observed that the two smallest communities clustered together, while the second largest community and the largest community were clustered on their own (based on the number of node colors for each block). I then assumed that only 2 communities existed. I saw that the largest community had the majority of nodes as one color in its block while the other color was spread across the nodes of the other 3 blocks, showing up as a majority in these 3 blocks.

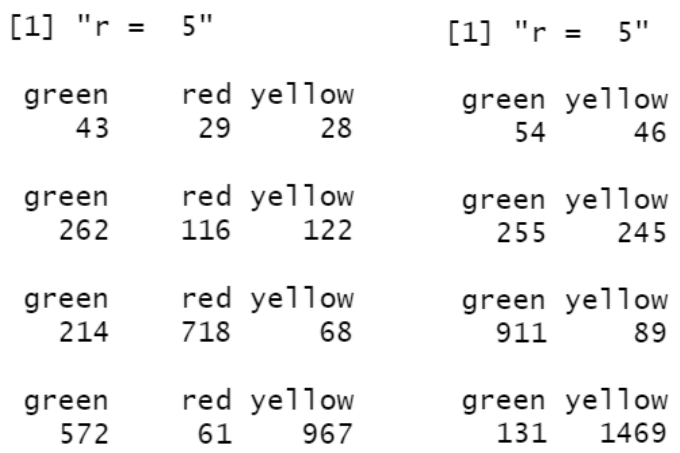

Figure 17: On the left is the contingency table of nodes per color in each block (total of 4 blocks) for the assumption of 3 communities. The majority of nodes in the last row (which indicates the 4th block) are yellow, the majority of nodes in the 2nd largest block are red, and the most frequent nodes in the two smallest blocks are green. On the right is the contingency table of nodes per color in each block for the assumption of 2 communities. The majority of nodes in the largest block row are yellow and the majority of nodes in the 3 smaller blocks are green. For this case, the within edge probabilities are 0.03 and the between edge probabilities are 0.015. The block sizes are 100, 500, 1000, and 1600. The *r* value here is 5, since clusters are more apparent with this larger *r* value.

From several simulations, extremely large *r* values typically do not yield dramatic increases in overlap nor modularity. There may sometimes be slight increases in clustering quality, but most of the time, the overlap or modularity plateaus and may begin to decline. There may also be fluctuations around a value, but not enough to sharply increase overlap or modularity.

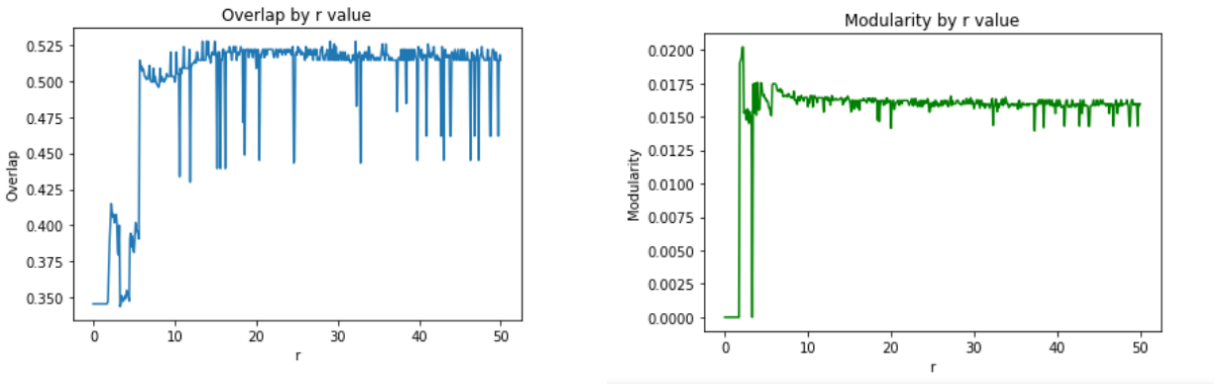

Figure 18: Overlap and modularity by *r* value for a graph with all edge probabilities of varying values ranging from 0.20 to 0.26 and unevenly sized blocks. With increasing *r* values, an overlap peak is actually reached at $r = 15$. However, from there the overlap fluctuates and dips near this peak value. The modularity peaks at an early value of *r* and declines with increasing *r* values.

5. Discussion

Finding patterns in how the value of *r* affects community detection with the Bethe-Hessian matrix formula can help us to find groups of related nodes more efficiently. Although utilizing very large *r* values may slightly improve overlap and modularity, they often lead to similar clustering results as those of small *r* values, and perhaps a drop in clustering efficiency. Therefore, choosing small *r* values tend to be ideal, and they are most often evident when one sees several of the graph nodes change color (indicating they have been clustered into a different community). As was seen with some of the real networks, it is possible to find an r value less than 1 (or less than -1 in the disassortative case) which maximizes overlap or modularity. However, one must take a closer look at how the nodes are clustered in these cases to see if it is truly representative of each community. A better algorithm for detecting the number of communities is an area of potential research for the future, as simply looking at the number of negative eigenvalues may not be ideal for communities that are not well-defined. $\sqrt{c}$ and $\sqrt{\rho(B)}$ [1] failed to be ideal choices of *r* in more sophisticated graphs. The Bethe-Hessian can be extended to weighted graphs by utilizing the weighted Bethe-Hessian matrix formula given in [1], along with very large graphs containing several communities. The results of this study can be further studied in graphs where the distribution of node degrees is heterogeneous as opposed to homogeneous.

6. References

[1] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. Spectral clustering of graphs with the bethe hessian. In Advances in Neural Information Processing Systems, pages 406–414, 2014.

[2] Arash A. Amini, Aiyou Chen, Peter J. Bickel, and Elizaveta Levina. Pseudo-likelihood methods for community detection in large sparse networks. Annals of Statistics, Vol. 41, No. 4, pages 2097-2122, 2013.

[3] Emmanuel Abbe. Community detection and stochastic block models: recent developments. Journal of Machine Learning Research 18, pages 1-86, 2018.

[4] Lorenzo Dall'Amico and Romain Couillet. Community detection in sparse realistic graphs: improving the bethe-hessian. IEEE, 2019.

[5] Lorenzo Dall'Amico, Romain Couillet, and Nicolas Tremblay. Revisiting the bethe-hessian: improved community detection in sparse heterogeneous graphs. In Advances in Neural Information Processing Systems, pages 4039–4049, 2019.

[6] Newman, Mark. "Network Data." *Network Data*, 19 Apr. 2013, www-personal.umich.edu/~mejn/netdata/.

[7] Piech, Chris. "K Means." *CS221*, Stanford, 2013, stanford.edu/~cpiech/cs221/handouts/kmeans.html.

[8] Santo Fortunato. Community detection in graphs. Physics Reports 486, pages 75-174, 2010.

[9] Ulrike von Luxburg. A tutorial on spectral clustering. Statistics and Computing 17(4), 2007.

7. Reflection

My experience conducting research has helped me grow as a person. I have faced and overcome several obstacles, learned new topics in a field I am interested in exploring further, and have developed better communication skills. My first task in starting my research project was to first look over research articles to gain an understanding of the topic I would be investigating. Understanding the background research was difficult at first, since I had never read published research articles in this field before. I realized I needed to read through the articles slowly and carefully, and research terms I did not understand to attempt to understand what the authors were discussing. Even if some of the research articles were not directly related to the research I performed for my project, I was able to learn more concepts and findings about network theory that may help me later on in my career.

As I am still somewhat new to statistical coding, conducting simulations with different types of networks and graphs in the programming language R was difficult at first. However, with the help of my faculty sponsor and researching online, I was able to overcome these difficulties and figure out more efficient ways of programming. I understood the need to use functions and loops instead of hard coding number manually, which I had a tendency to do. My faculty sponsor pointed this out to me, and I was able to figure out ways to improve my code to make it faster. I learned new tools, new functions, and new ways of presenting data in the form of graphs, histograms, and tables. I found that I could conduct simulations in a cleaner and more presentable way in Python, so I started to utilize Python over R to continue my research. I had never used Python before for statistical programming, but I learned there were several similarities between R and Python, which helped speed up the learning curve. Debugging code

sometimes took hours and at times was frustrating, but when I finally figured out the problem and the solution, I felt a sense of accomplishment.

I understood that research is about being explorative and curious. It is different from a homework assignment where one finds the solution to a question and moves onto the next question. Rather than studying for an upcoming exam, research is about actively learning new techniques and applying these techniques to help gain a better understanding of why things behave the way they do. In research, one must look for anomalies or patterns in the data and try to understand what could be the potential reason behind these anomalies and patterns. In this sense, conducting research was a nice change from the usual assignments and exams associated with undergraduate courses.

Conducting research also allowed me to utilize some of the knowledge and tools that I have learned from previous courses I have taken at UC Davis. For example, some of the previous programming I had done with R in my Statistical Data Science course (STA 141A) was applicable to running simulations and analyzing data. Even some of the concepts from my computer science courses (ECS 10 and ECS 30) were crucial to helping me code. In terms of statistical concepts and theory, the knowledge I gained from my Statistical Learning course (STA 142B) was helpful for me to understand what machine learning really is about and why it useful to help in data analysis and data prediction. The Bethe-Hessian algorithm is indeed a machine learning algorithm that can be used in the context of networks and graphs to detect communities. I was also able to apply the eigenvalue and eigenvector concepts associated with matrices, which I learned from my linear algebra courses (MAT 22A and MAT 167).

In-person meetings at the Mathematical Sciences Building along with Skype video meetings with my faculty sponsor helped me develop my communication skills. In data science, I realize that explaining and interpreting results are crucial. If one cannot explain the information contained in the data and what he or she has found, then it is not very useful and meaningful to others. Although I struggled initially in communicating my ideas and work with my faculty sponsor, I feel that I have improved thanks to his guidance and the experience associated with having multiple meetings. I also think there is room for me to grow in this area.

Presenting my work at the 2020 UC Davis Undergraduate Research Conference also provided me with an opportunity to improve my communication skills. It gave me an opportunity to present to viewers who had no prior experience in this field of research. I had to consolidate my research background and findings into a short eight minute video, and I had to speak clearly into a camera. During the research conference, I also responded to questions from conference viewers. This experience made me more confident in my presentation skills and my ability to answer questions from others.

Overall, I am very happy I was able to get a chance to conduct research related to machine learning and develop a relationship with a faculty sponsor from the Statistics department. I think the Bethe-Hessian can be a powerful machine learning algorithm to help network scientists detect communities within graphs, and I believe my research has helped us to get a better understanding of the algorithm. This experience has only heightened my interested in data science and machine learning, and I hope that I can explore these concepts further in the future, as well as utilize the skills I have acquired to be successful in my career.