

Application of Graph Theoretic Clustering on Some Biomedical Data Sets

by Darla Ahlert, Bachelor of Science

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Degree of
Master of Science
in the field of Computer Science

Advisory Committee:

Gunes Ercal, Chair

Igor Crk

Mark McKenney

Graduate School
Southern Illinois University Edwardsville
May, 2015

UMI Number: 1588658

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1588658

Published by ProQuest LLC (2015). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

ABSTRACT

APPLICATION OF GRAPH THEORETIC CLUSTERING ON SOME BIOMEDICAL DATA SETS

by

DARLA AHLERT

Chairperson: Dr. Gunes Ercal

Clustering algorithms have become a popular way to analyze biomedical data sets and in particular, gene expression data. Since these data sets are often large, it is difficult to gather useful information from them as a whole. Clustering is a proven method to extract knowledge about the data that can eventually lead to many discoveries in the biological world. Hierarchical clustering is used frequently to interpret gene expression data, but recently, graph-theoretic clustering algorithms have started to gain some attraction for analysis of this type of data. We consider five graph-theoretic clustering algorithms run over a post-mortem gene expression dataset, as well as a few different biomedical data sets, in which the ground truth, or class label, is known for each data point. We then externally evaluate the algorithms based on the accuracy of the resulting clusters against the ground truth clusters. Comparing the results of each of the algorithms run over all of the datasets, we found that our algorithms are efficient on the real biomedical datasets but find gene expression data especially difficult to handle.

TABLE OF CONTENTS

ABSTRACT	ii
LIST OF FIGURES	v
LIST OF TABLES	vii
Chapter	
1. INTRODUCTION	1
2. RELATED WORK	6
3. CLUSTERING ALGORITHMS	10
3.1 K-Means	10
3.2 Cluster Affinity Search Technique	11
3.3 Vertex Attack Tolerance and Hierarchical Vertex Attack Tolerance	13
3.4 Girvan-Newman	14
4. DATA AND EXPERIMENTS	16
4.1 Gene Expression	16
4.2 Preprocessing and Clustering	16
4.2.1 Biclustering	18
4.3 Datasets	19
4.3.1 Post-mortem gene expression data	19
4.3.2 Differentially expressed genes	20
4.3.3 Real datasets	21
5. IMPLEMENTATION AND RESULTS	23
5.1 Implementation	23
5.1.1 Scikit-learn	23
5.1.2 GUI	24
5.2 Result Validation	25
5.2.1 Error automation	27
5.3 Results	30
5.3.1 Clustering the real datasets	31
5.3.2 Clustering the post-mortem data using all genes	33
5.3.3 Using differentially expressed genes	35
5.3.4 Looking into the details of the results	40
5.3.5 An attempt at biclustering	43

5.3.6 Discussion of results and contributions	43
6. CONCLUSIONS AND FUTURE WORK	46
REFERENCES	50

LIST OF FIGURES

Figure	Page
1.1 Hierarchical Dendrogram	2
1.2 Weighted Graph	3
3.1 k -Means Algorithm	11
3.2 CAST Algorithm	12
4.1 Gene Expression Matrix	17
5.1 Spectral CoClustering Result	24
5.2 Spectral BiClustering Result	24
5.3 Graphical User Interface	25
5.4 Cluster Tab	26
5.5 Example <i>.cluster</i> File	27
5.6 Data File Containing the Correct Clustering	28
5.7 Hierarchical VAT-Clust Visualization of Iris Dataset	33
5.8 GEO Graph of <i>AllSamplesAllGenes</i> Dataset	35
5.9 GEO Graph of <i>CvBP</i>	37
5.10 GEO Graph of <i>CvS</i>	37
5.11 BiPart_BPGenes <i>.cluster</i> File	41
5.12 Venn Diagram Showing Differential Gene Numbers	41
5.13 CvBP <i>.cluster</i> file	42

5.14	CvS <i>.cluster</i> file	42
------	------------------------------------	----

LIST OF TABLES

Table		Page
4.1	Summary of Subjects in Post-Mortem Dataset	20
4.2	Summary of All Datasets	22
5.1	Wine K -means Results	29
5.2	Real Data Results	32
5.3	Results Using All Genes	34
5.4	Results Using Bipolar Genes	36
5.5	Results Using Schizophrenia Genes	38
5.6	Comparison with All Samples	39
5.7	Control vs. Other Samples	40
5.8	Biclustering Results	44

CHAPTER 1

INTRODUCTION

Take a minute to think about the amount of data that has been collected in the world up to this point in time. There is a good chance that a million (or more!) pieces of data have already been gathered just in the extremely short amount of time that it took to read that sentence. Data is everywhere. One thing that is common among many datasets is that different points tend to group together based on some metric of similarity. This is helpful in data analysis because it is in our human nature to attempt to classify things into certain categories [2]. This grouping of data points is called *clustering*. By clustering these points we are able to notice trends or patterns that are associated with a dataset, which can be of great help in finding crucial facts about the data.

Clustering is a popular technique for analyzing data in many fields, such as computer science, engineering, and biology. Generally, each point, x , in the data is represented as a t -dimensional vector $x = \langle a_1, a_2, \dots, a_t \rangle$ where each dimension represents a feature that is common among all of the data points [27]. Each dimension is then represented by a real number specific to its data point. We can cluster these points based on their similarities and differences such as how close or far apart they are from one another. If there is a data set with N points that need to be clustered, a similarity matrix of size $N \times N$ can be constructed by calculating the similarity between each of the N points. The entries in the matrix can be determined using many different techniques. A common technique used to measure similarity is the Euclidean distance between each point, but some other options include the Manhattan distance, Canberra distance, or the Spearman correlation [15]. Once the similarity matrix has been constructed, the clustering can begin.

Hierarchical clustering is a common clustering technique used to cluster biological data [16]. In general, this type of clustering will organize the data into a hierarchical structure

based on the similarity between data points. This is often depicted by a dendrogram, which can be seen in figure 1.1 [27]. We can see that at the top of the dendrogram all of the data points have been placed into one large cluster. As we travel down through the dendrogram the points are broken up into smaller and smaller clusters. Depending on the number of appropriate clusters, it is possible to essentially cut off a portion of the bottom and be left with the amount of clusters desired [15].

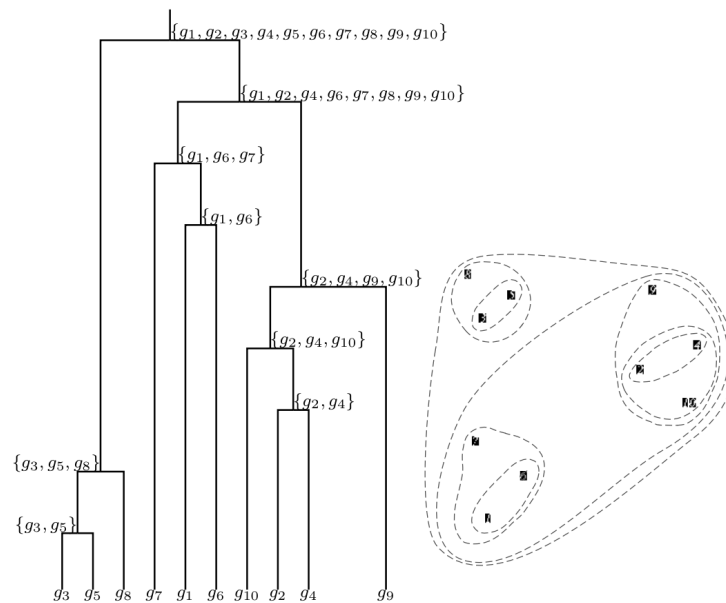


Figure 1.1: Hierarchical Dendrogram

Recently, there has been more interest in using graph-theoretic clustering algorithms to analyze biological data. Given input data, a weighted graph G can be constructed where the nodes V correspond to the data points being clustered, and the edges E correspond to the distances or similarities between the data points. A very simple example of a weighted graph that we put together can be seen in figure 1.2. These graphs can be constructed in a few different ways. One way is the k -nearest neighbor graph that is constructed by only allowing each node to have k neighbors [22]. These neighbors are chosen by selecting the k closest nodes to the current node. Another common construction is the minimum

connectivity graph. This graph is constructed with all nodes and the minimum number of edges needed to keep the graph connected (i.e. there is a path to and from each node to all other nodes).

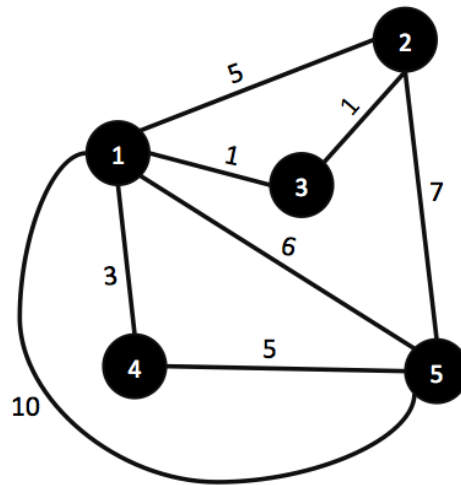


Figure 1.2: Weighted Graph

Using the given data and resulting graph, we wish to cluster the data points in a way in which the points belonging to a cluster are extremely similar and those belonging to different clusters are very dissimilar. Since the Euclidean distance is often used as the similarity measure, the goal would then be for the data points within a resulting cluster to be very close to each other while data points in separate clusters are far apart [23]. This seems straightforward at first, but once the amount of variables for clustering is discovered it quickly becomes much more complicated. Different graph constructions as well as different similarity measures will most definitely lead to different results [1]. Also, even if the Euclidean distance is being used across the board, the way in which each algorithm finds or defines a cluster can vastly change the output. There has been no universal clustering algorithm shown to solve all problems, which makes it crucial to explore many different clustering algorithms and research which might be best for the data at hand. On top of all of the potential clustering algorithms to choose from, each algorithm often

has many different parameters that help determine a final clustering. Most clustering explorations attempt these algorithms with varying values for the parameters. This allows the researcher to see which parameters fit the data best [22].

With all of this in mind, we can move on to biomedical datasets. It will be shown later in this paper that these datasets can be a bit tricky. The experiments behind these sets of data, or how the data was gathered, are often different for each dataset. We have found that gene expression data in particular is undoubtedly difficult to deal with. It often has very few rows but thousands of columns, making for extremely high-dimensional data. For our research, we have chosen to explore and analyze one gene expression dataset consisting of data pulled from post-mortem brains. We also explore some real datasets gathered from the UCI Machine Learning Repository. These real datasets have clusters already known and classified within the data that allow us to see how well our chosen algorithms run on different types of data.

The main goal of this research is to squeeze the tool of clustering on biomedical datasets and, specifically, gene expression data. We want to be able to answer questions such as:

- How do different clustering algorithms compare to each other on biomedical datasets?
- Does one clustering algorithm always give the highest accuracy for all scenarios?
- How well does our new clustering algorithm, VAT-Clust, and different variations of VAT-Clust, work on different biomedical datasets?
 - Which type of graph constructions give the best accuracy?
 - Which variation(s) of VAT-Clust give the best accuracy?
 - How does VAT-Clust compare to the different clustering algorithms?
- In regards to gene expression data,

- What is the best preprocessing technique?
- What is the effect of restricting ourselves to only particularly important genes?

The answers to these questions are unclear. It is our hope to be able to answer these questions by the end of this research. Chapter 2 discusses some of the related work in this area. Chapter 3 explores the five different clustering algorithms we chose to implement. Chapter 4 deals with the details of the data and experiments we completed. Chapter 5 discusses the actual implementations and tools we used as well as how we calculate the accuracy of a given clustering algorithm. It also discusses and analyzes the results obtained from running the algorithms in Chapter 4 over the data in Chapter 5. Finally, Chapter 6 concludes the paper and discusses some potential future work.

CHAPTER 2

RELATED WORK

The application of clustering data has been studied across a wide variety of subjects. Clustering gives the researcher the ability to see trends in data without having any prior knowledge of how the data may be constructed or connected. Because so many different fields have plunged into clustering research, a massive amount of clustering algorithms have been created. Xu and Wunsch discuss over 35 clustering algorithms in their paper alone [27]. Even after narrowing the research down to clustering techniques of gene expression data, it still is not very clear which algorithms are the best. This stems from the fact that finding meaningful trends and patterns in gene expression data is non-trivial. In fact, many of the widespread approaches fail to successfully cluster this difficult data. Gene expression data is often large and contains a lot of data that is not necessarily important to the functionality of a specific gene or particular gene functions. Essentially every paper exploring the clustering of gene expression data says that some sort of preprocessing or feature selection is absolutely necessary as an attempt to get rid of all this nonsense data and for these clustering algorithms to perform well [16, 14, 26, 1].

The concept of preprocessing has become a rather large research topic in and of itself. Just as there are many different clustering algorithms to choose from, there are also many different preprocessing techniques that can be used on data. One technique is to normalize the data. By finding the mean of each row and subtracting that value from only the elements in that row we consider the row normalized. This can be done with different types of means. Other types of preprocessing include taking the logarithmic transformation of the data or selecting only genes that are found to be significantly different from the rest of the data as a whole (known as feature selection) [14]. Similar to selecting only certain genes through feature selection, feature extraction makes it

possible to actually transform the given genes into a new, smaller dataset that represents a simplified version of the data. This can be done through principle component analysis [25].

Along with selecting a preprocessing technique, it is also necessary to select a similarity measure. Similarity can be measured by some sort of correlation measure or even different distance measures. The Pearson correlation, for example, is a fairly well known correlation. This correlation tends to work very well when there is a strongly linear relationship between the two objects being observed but can be sensitive to outliers [1]. The other option to define similarity is some sort of distance measure. The most popular distance measure is the Euclidean distance, which can be seen in equation 2.1. Another popular distance measure is the Manhattan distance, often referred to as the city-block distance, which can be seen in equation 2.2.

$$d(x, y) = \sqrt{\sum_{k=1}^K (x_k - y_k)^2} \quad (2.1)$$

$$d(x, y) = \sum_{k=1}^K |x_k - y_k| \quad (2.2)$$

Different similarity measures will result in different clusterings even from the same algorithm. Research in gene expression clustering has so far guided people to stick with the most popular option, Euclidean distance [14, 1, 16].

Now that it is possible to preprocess data and choose how to measure similarity, the actual clustering can begin. Just as there is no universal clustering technique for all types of data, there is also no universal technique for clustering just gene expression data. Actually, it is still pretty unclear as to which algorithm works best overall in this scenario. The first thing to consider is whether there will be gene-based clustering or sample-based clustering. The two names are self-explanatory. With gene-based clustering we are attempting to cluster the genes into groups, and with sample-based clustering we

are attempting to cluster the samples into groups [14]. Both have different reasonings behind them. For example, sample-based clustering is often used when the researcher wants to identify specific groups of samples. Say you are testing a group of people that consist of three subgroups: one group is the control group, one group is a group that has a disease, and one group is a group that has a disease but is being treated with some medication. These subgroups can potentially be found with a sample-based clustering approach. With gene-based clustering, researchers are often trying to find genes that work together during certain processes. Finding how and when these genes work together can result in extremely important discoveries in the world of genetics. Jiang et al. analyzes both of these methods [14].

Whether gene- or sample-based clustering, the algorithm itself can be chosen from the many different ones available. As stated before, Xu and Wunsch discuss and dive into the details of a very large amount of clustering algorithms. Jiang et al., as well as Kerr et al. consider numerous clustering algorithms in regards to just gene expression data [14, 16]. A final approach popular in gene expression clustering is when both the genes and samples are clustered at the same time. This has many different names such as biclustering, co-clustering, two-way clustering, subspace clustering, etc. [10, 6, 21, 17]. This type of clustering essentially finds submatrices within the data matrix. These submatrices result in subsets of genes that work together over a subset of samples, which is exactly what is needed.

Clearly, there are many different variables that go into the study of clustering. A final important component in clustering is the validation of the resulting clusters. With all of the different preprocessing, algorithms, and parameters for the algorithms, the resulting clusterings are bound to be very different for each combination of variables. Cluster validation is a way of testing the quality of the clusters that are output. This can be done internally or externally depending on if there is prior knowledge of the data. If the

actual clusters are known and the researcher is attempting to replicate those clusters, then external validation is appropriate. The researcher would simply compare their result clusters to the given clusters. If there are no known “ground truth” clusters, then internal validation is appropriate [27]. More details of cluster validation can be read in Chapter 5.

CHAPTER 3

CLUSTERING ALGORITHMS

3.1 K-Means

The k -means algorithm is a widely known and frequently used algorithm. There are a few slightly different versions of the k -means algorithm, but the one we implemented is known as Lloyd's algorithm. Given $M = \{m_1, \dots, m_n\}$ data points in d -dimensional space, the goal is to partition the data into k groups (or clusters) while minimizing the sum of the squared error of the clusters. The algorithm itself is very simple. The initial step is to randomly pick k nodes to be considered centroids of the data set $X = \{x_1, \dots, x_k\}$. A data point m is added to a cluster if adding that data point minimizes the squared error distortion of the given cluster. This can be calculated given the following formula:

$$d(M, X) = \frac{\sum_{i=1}^n d(m_i, X)^2}{n} \quad (3.1)$$

where $d(m, X) = \min_{1 \leq i \leq k} d(m, x_i)$. The distance $d(m, x_i)$ can be calculated as one of many different distance measures, but is most commonly calculated as the Euclidean distance. This process (minus the initialization step) is repeated until there is no change for each cluster. The algorithm can be seen in figure 3.1.

The time complexity of the k -means algorithm is $O(M k d j)$ where j is the number of iterations. Since k and d are usually fairly small, the algorithm will run quickly even on larger data sets [27]. The problem with this algorithm is that unless you somehow know k in advance, there really is not a way to identify what k should be given only the data. Also, outliers and noise can affect the quality of clustering that k -means returns. Even though it is the goal to minimize the squared error of the clusters, every node must still be added to one of the clusters. Therefore, outliers and noise will be clustered and

thus, skew the shape of each cluster [27].

```

PROGRESSIVEGREEDYK-MEANS( $k$ )
1  Select an arbitrary partition  $P$  into  $k$  clusters.
2  while forever
3       $bestChange \leftarrow 0$ 
4      for every cluster  $C$ 
5          for every element  $i \notin C$ 
6              if moving  $i$  to cluster  $C$  reduces the clustering cost
7                  if  $\Delta(i \rightarrow C) > bestChange$ 
8                       $bestChange \leftarrow \Delta(i \rightarrow C)$ 
9                       $i^* \leftarrow i$ 
10                      $C^* \leftarrow C$ 
11     if  $bestChange > 0$ 
12         change partition  $P$  by moving  $i^*$  to  $C^*$ 
13     else
14         return  $P$ 

```

Figure 3.1: k -Means Algorithm

While these characteristics are undesirable, there are ways to work around them. Since the first k centers are chosen at random, the resulting clusters could potentially be different each time. The best option is to run the algorithm multiple times for many different k values. This will give a well-rounded “average” of resulting clusters for each k value which can then be analyzed to find the which of these k values is the best fit for the data [14]. This seems like a lot of work, but in reality the simplicity of the implementation outweighs, or at worst counterbalances, the extra work put in.

3.2 Cluster Affinity Search Technique

The Cluster Affinity Search Technique (CAST) algorithm created by Amir Ben-Dor and his colleagues is a practical heuristic that is often used as a clustering algorithm on gene expression data. CAST clusters a set of vertices S according to a given distance graph G and threshold distance value Θ . The distance between a gene i and a cluster C is defined as the average distance between i and all other genes in C :

$$d(i, C) = \frac{\sum_{j \in C} d(i, j)}{|C|} \quad (3.2)$$

The algorithm (shown in figure 3.2 [15]) will iteratively build a partition P of the set S by creating a candidate cluster C in which no gene i not belonging to C is within distance Θ , and that no gene i belonging to C is not within distance Θ . Once a cluster is stable (i.e. no removal or addition of new nodes will occur), the current cluster C will be added to the partition P . The nodes belonging to C will then be removed from the set of vertices S and C will be set to an empty cluster. The algorithm will continue to build clusters until the set S is empty [15].

```

CAST( $G, \theta$ )
1   $S \leftarrow$  set of vertices in the distance graph  $G$ 
2   $P \leftarrow \emptyset$ 
3  while  $S \neq \emptyset$ 
4       $v \leftarrow$  vertex of maximal degree in the distance graph  $G$ .
5       $C \leftarrow \{v\}$ 
6      while there exists a close gene  $i \notin C$  or distant gene  $i \in C$ 
7          Find the nearest close gene  $i \notin C$  and add it to  $C$ .
8          Find the farthest distant gene  $i \in C$  and remove it from  $C$ .
9      Add cluster  $C$  to the partition  $P$ 
10      $S \leftarrow S \setminus C$ 
11     Remove vertices of cluster  $C$  from the distance graph  $G$ 
12 return  $P$ 

```

Figure 3.2: CAST Algorithm

CAST has no performance guarantee but is known to work extremely well with gene expression data [3]. In fact, it is possible that CAST may never converge on its final cluster. For this reason, a check has been added to our implementation of the algorithm. If the remaining unclustered nodes are not within distance Θ to any of the clusters, we consider them unclustered outliers and stop the loop of the algorithm. Although this situation may occur, CAST is a good choice as a clustering algorithm because it does not depend on the user to define a number of clusters. Therefore, it seems as though CAST

has the ability to find the underlying structure of the data. Also, because of the distance threshold, CAST does not force outliers into any cluster like the k -means algorithm will [14]. Regardless, there is still the issue of choosing a good value for Θ .

3.3 Vertex Attack Tolerance and Hierarchical Vertex Attack Tolerance

Originated by Gunes Ercal, John Matta, and Jeff Borwey the Vertex Attack Tolerance (VAT) is a node-based resilience measure that can be used to cluster data in a graph-theoretic manner. A graph $G = (V, E)$ is defined by a set of vertices, V , and a set of edges, E . The resilience of a graph indicates the graph's ability to handle an attack, or removal of some edges or nodes. This can often be found by first finding the conductance of a graph. This involves finding the edge or set of edges that would cause the biggest disconnection in the graph, known as the critical edge set [9]. With VAT, the resilience measure considers vertex-based attacks instead. It represents the smallest number of nodes that must be removed to disconnect the given graph G and can be defined as

$$\tau(G) = \min_{S \subset V} \frac{|S|}{|V - S - C_{max}(V - S)| + 1} \quad (3.3)$$

where $C_{max}(V - S)$ is the largest component remaining in the graph upon the removal of critical node set S [23]. The removal of S separates the graph into multiple connected components that become the candidate clusters for the data set. For a full clustering of the data, we use the VAT-clust algorithm where each of these critical nodes gets assigned to a cluster in a greedy manner based on the maximum number of neighbors the node will have when added back into the graph [5]. Like CAST, VAT-Clust does not need the user to specify the number of clusters the data will have which comes in handy when k is not known.

Another version of VAT-Clust, called Hierarchical VAT, can be used to guarantee a minimum number of clusters k (taken as a parameter) from the original graph G_0 . It

begins by running VAT on G_0 . If the number of resulting clusters is at least k , it will converge. Otherwise, new graphs $\{G_C\}$ are created from the resulting clusters C , and VAT values are computed for each of the new graphs, G_C . VAT is then recursively run on the graph that has the minimum VAT value to get new separate, connected components. This process is repeated until the numbers of total clusters is at least k . Hierarchical VAT is nice because it will guarantee at least k clusters. So, if there exists a priori knowledge of the graph structure it is possible to somewhat force the results into at least the correct number of clusters.

3.4 Girvan-Newman

The Girvan-Newman algorithm is a graph-theoretic algorithm that focuses on the betweenness of a graph. The betweenness centrality of a node i is defined as the number of shortest paths between two other nodes that must travel through i to get to one another. This measure represents the importance of i in keeping the flow between nodes in a graph. We can define *flow* as the ability to transfer information between nodes in a graph. Betweenness centrality can also be calculated on an edge basis, where the definition is the same but instead is concerned with an edge e . Higher values of betweenness centrality for e mean that the edge is important in keeping the graph's structure and flow [11]. With edge betweenness centrality in mind, Michelle Girvan and M. E. J. Newman proposed an algorithm in which the betweenness is calculated for all edges in the graph, and the edges with the highest values get removed from the graph. This removal of edges will eventually lead to separate, connected components that are considered the candidate clusters of the graph. The algorithm is fairly simple:

1. Calculate the betweenness for all edges in the network
2. Remove the edge with the highest betweenness.
3. Recalculate the betweenness for all edges affected by the removal.
4. Repeat steps 2 and 3 until no edges remain.

Considering m edges and n nodes, the worst-case time complexity of the algorithm is $O(m^2n)$ since the betweenness calculation has to be repeated each time an edge is removed. Having said that, the calculation only has to be computed for those edges that were affected by the removal, which is at most only the edges in the one connected component that contained the edge that was removed. So, the running time will often be much better than this worst-case time described above.

CHAPTER 4

DATA AND EXPERIMENTS

4.1 Gene Expression

All living organisms go through certain biological processes that are vital for the organism to stay alive. During these processes, different genes are expressed, or active, at different times and at different levels. These levels can be higher or lower than normal (upregulated or downregulated respectively). When this occurs, it can be said that the gene is differentially expressed. Based on the level and timing of differential expression, it is possible to see how significant the specific gene is to the current biological process [3]. DNA Microarray technology has brought the field of genetics to a new level. With this technology, it is now possible to measure the levels of expression for thousands of genes simultaneously during certain biological processes. The gene expression levels cannot only be measured for thousands of genes in one sample, but across many samples at once. This data can be significant in finding insight into certain gene functions or how genes work with one another [14]. Gene expression data is often represented by an $n \times m$ matrix $M = \{w_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$ containing real values where the rows $G = \{\vec{g}_1, \dots, \vec{g}_n\}$ represent the gene expression patterns of particular genes, and the columns $S = \{\vec{s}_1, \dots, \vec{s}_m\}$ represent particular samples and their expression profiles. Therefore, each cell's real number value w_{ij} represents the expression level of gene i in sample j . This can be visualized in figure 4.1 [14].

4.2 Preprocessing and Clustering

Given a gene expression data matrix as described above, we can group objects together into clusters so that objects within a cluster are very similar while objects in different clusters are very dissimilar. Gene expression data can be clustered on a gene basis, on a

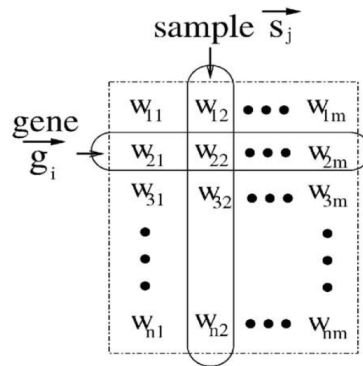


Figure 4.1: Gene Expression Matrix

sample basis, or even a combination of the two. In gene based clustering, the genes are considered the objects with the samples being the features. Similarly in sample-based clustering, the samples are considered the objects with the genes being the features. Finally, when clustering both genes and samples (i.e. two-way clustering or biclustering), both the genes and the samples can be considered objects or features. No matter which clustering method is being used, it is wise to conduct some sort of preprocessing on the data. There are many different techniques. A couple of examples include performing the logarithmic transformation on each w_{ij} value or standardizing each row to have a mean of zero and a standard deviation of one. Each of the different clustering methods might require different preprocessing techniques, and from our experience, which preprocessing procedure to carry out to get the best results is by no means trivial to find. We do know that one important preprocessing technique in sample-based clustering is to somehow find the genes that are relevant to the samples. Previous research shows that less than 10% of the genes in a given gene expression matrix will be important or *informative* [26]. This means that gene expression data is extremely *noisy* (i.e. a lot of data exists that does not contribute to the final clustering). Noise is caused by the complex procedures of microarray experiments [14]. Non-informative genes can have a very negative impact on certain clustering algorithms depending on how the algorithm is set up. For example, the

k -means algorithm mentioned in Chapter 3 clusters all objects regardless of how skewed the clusters become as a result of these outliers, or pieces of data that do not necessarily belong. Another major issue with gene expression data is the high-connectivity of it. If a graph is highly connected, then there are a lot of edges between all of the nodes. This can affect the results of an edge-based clustering algorithm, like the Girvan Newman algorithm listed in Chapter 3. This massive amount of edges can also contain noise. Although two nodes are connected by an edge, they might not necessarily belong in the same cluster. The existence of this edge could cause the algorithm to potentially think that they belong together and incorrectly cluster them.

All of the clustering algorithms in existence have strengths and weaknesses. Some clustering algorithms might work extremely well with certain datasets but perform poorly with others. Finding the “right” or “best” clustering algorithm altogether is extremely difficult. That is why it is very common to try many different clustering algorithms to see which best fits the data being processed. Clustering gene expression data is just another example of this, and this is exactly what we will attempt to do.

4.2.1 *Biclustering*

The algorithms discussed in Chapter 3 consider a clustering based on the rows of a given data matrix. Another approach that is mentioned above, known as *biclustering*, allows the rows and columns of a data matrix to be simultaneously clustered [10]. Biclustering was first introduced by J. A. Hartigan in 1972 as ‘*Direct Clustering*’ [12]. Eighteen years later Yizong Cheng and George M. Church applied it to gene expression data and called it *biclustering* [6]. Since then, biclustering has become a popular way to discover patterns in gene expression data, and many different types of biclustering algorithms have been introduced. One positive about biclustering is that it gives the user the ability to allow the rows and columns to belong to more than one bicluster. This is especially important for gene expression data because it is well known that a single gene may be

active under many different conditions. A gene can be a part of many clusters or even none at all. One downfall to biclustering is that finding an exact solution has been proven to be *NP*-hard. Therefore, the many different biclustering techniques all use some sort of heuristic approach to approximate the optimal clustering of a given dataset [10].

4.3 Datasets

We experiment with many different datasets. Based on some of our results and prior research, we also experiment with certain subsets of the given datasets. All of the experimental sets are described in detail in the subsections below. Table 4.2 also contains information regarding each of the datasets. The first column contains the short hand name for each dataset. The second column contains the number of known clusters. The third column contains the number of data points for each sample. The fourth column contains the total number of samples. Finally, the fifth column contains a brief description of each set.

4.3.1 Post-mortem gene expression data

The gene expression dataset¹ that caught our attention was found online in the Gene Expression Omnibus provided by the National Center for Biotechnology Information [8]. This dataset contains gene expression profiles of postmortem brains with major mental disorders such as bipolar disorder, depression, and schizophrenia. There is also a control group with no major mental disorder noted. The samples gathered were diagnosed according to the Diagnostic and Statistical Manual of Mental Disorders, Fourth Edition and details of each can be seen in table 4.1 [13].

This data matrix contains 50 total samples and 12,518 genes. The 50 samples consist of 15 control subjects, 11 samples with bipolar disorder, 11 samples with major depression, and 13 samples with schizophrenia. For each of these classifications the average age in

¹It is important to note that this data was collected and made publicly available by someone other than ourselves. We did not gather any information or do any human research.

	N	Age (<i>years</i>)	Gender	PMI (<i>hours</i>)
Control Subjects	15	48 ± 11	9M : 6F	24 ± 10
Bipolar Disorder	11	39 ± 12	8M : 3F	32 ± 16
Major Despression	11	46 ± 10	6M : 5F	27 ± 12
Schizophrenia	13	44 ± 14	8M : 5F	33 ± 15

Table 4.1: Summary of Subjects in Post-Mortem Dataset

years, number of males and females, and post-mortem interval (the number of hours that had passed since death when the samples were taken) is shown. Since there is no ground truth known for a correct clustering of the genes, any clusterings done on this dataset were on a sample basis. It is clear which samples belong to each category and thus, it is easy to calculate the accuracy of the resulting clusters. We give this dataset the name *AllGenesAllSamples* for short.

4.3.2 Differentially expressed genes

As discussed before, finding the *informative* genes in a given dataset is an extremely important part of the process. Genes that are differentially expressed for certain classifications have the potential to help identify the underlying clusters of samples and therefore, can be informative. Iwamoto et al. were able to find genes whose differential expression correspond to bipolar disorder using a software called Affymetrix [13]. Looking through the entire post-mortem dataset, we were able to find 51 of the 53 differentially expressed genes listed in their work. We created a subset of data, called *AllvsBP_BPGenes*, that contains all of the samples and only these given genes. The idea behind this is that if these genes are truly informative and can help identify a certain classification or sample, then the clustering algorithms should find the correct clustering more easily. At minimum, it seems that these genes should at least help to correctly cluster samples corresponding to the expressed classification. Thus, we should get better accuracy by using just this small subset of samples.

Along with this, we thought it would be interesting to see if we could separate just the bipolar samples from all of the other samples or even just the control samples by using these differentially expressed genes in the clusterings. We created three more subsets of the post-mortem data. One contains only the control subjects and bipolar disorder samples, called *CvBP_BPGenes*. The other, called *BiClust_BPGenes*, contains all of the samples, but with the control subjects, schizophrenia, and depression samples all as one big group and bipolar as a second, smaller group. Finally, just to see what kind of results we could get, we created one last subset containing only the control subjects and bipolar disorder samples, but used all of the genes in the post-mortem dataset as opposed to just the differentially expressed genes. We call this set *CvBP_AllGenes*. Since Iwamoto et al. focused only on finding differentially expressed genes for bipolar disorder, we thought it would be interesting to attempt to find differentially expressed genes for schizophrenia and major depression as well. We were successful at finding research on differentially expressed genes for schizophrenia but not major depression. Collins et al. focus on one target gene in particular, but provide a list of differentially expressed genes for schizophrenia in their supplementary materials [7]. Out of the 202 genes on their list, we were able to find 100 in the post-mortem dataset. Using the schizophrenia genes, we create four similar subsets as described above for bipolar disorder. These have the same naming conventions and can be seen in table 4.2.

4.3.3 Real datasets

Finally, we want to test how well these algorithms run on biological data other than gene expression data. Thanks to the UCI Machine Learning Repository containing 311 datasets, we were able to find what we needed [19]. Out of the 311, we pick five popular biological datasets (Breast, Breast Wisconsin, Ecoli, Iris, and Wine) from the repository. Other than the fact that they are popular there is really no reason for choosing these five versus any other five. Although, these five datasets have different underlying structures

Dataset	Clusters	Dimensions	Samples	Description
Wine	3	13	178	Wines derived from three different cultivars
Iris	3	3	150	Three types of iris plants
Breast	6	9	106	Breast Tissue data
Breast_W	2	30	569	Breast data pulled from WI
Ecoli	8	7	336	Ecoli data
AllGenesAllSamples	4	12518	50	All samples & all genes
CvBP_AllGenes	2	12518	26	Control subjects / bipolar samples & all genes
CvBP_BPGenes	2	90	26	Control subjects / bipolar samples & bipolar genes
AllvsBP_BPGenes	4	90	50	All samples & bipolar genes
BiPart_BPGenes	2	90	50	Bipolar samples vs everything else
CvS_AllGenes	2	12518	28	Control subjects / schizophrenia samples & all genes
CvS_SchGenes	2	135	28	Control subjects / schizophrenia samples & schizophrenia genes
AllvSch_SchGenes	4	135	50	All samples & schizophrenia genes
BiPart_SchGenes	2	135	50	Schizo samples vs everything else

Table 4.2: Summary of All Datasets

as well as varying amounts of data associated with them, giving us a wide variety of data to experiment with. These five datasets are also listed in table 4.2.

CHAPTER 5

IMPLEMENTATION AND RESULTS

Combinations of our own implementations as well as others' implementations were used to run experiments and calculate result accuracy throughout this research. We describe these implementations and the resulting accuracies below.

5.1 Implementation

Throughout this research we were able to find readily available tools, like scikit-learn, that helped make this process much easier. We also had the help of another student's project to automate certain steps and gather almost all of our algorithms and tests into one central graphical user interface (GUI).

5.1.1 Scikit-learn

One of the tools used in this research was *scikit-learn*. This module, written in Python, integrates many different machine learning algorithms and tools. It is publicly available (<http://scikit-learn.org>) and can be found easily with a quick search on the Internet. *Scikit-learn* has tools for classification, clustering, dimensionality reduction, data preprocessing, and model selection [24]. Of these, we took advantage of the biclustering submodule. This submodule contains two different biclustering algorithms. Since different biclustering algorithms define biclusters differently, the resulting clusters will potentially have different structures. The first biclustering algorithm, called *Spectral Co-Clustering*, allows each row and each column to belong to only one bicluster. This will result in a diagonal type structure as shown in figure 5.1. This second biclustering algorithm, called *Spectral Biclustering*, will result in a checkerboard type structure. This result comes from the ability to allow each row and each column to belong to more than one final bicluster. The checkerboard structure can be seen in figure 5.2.

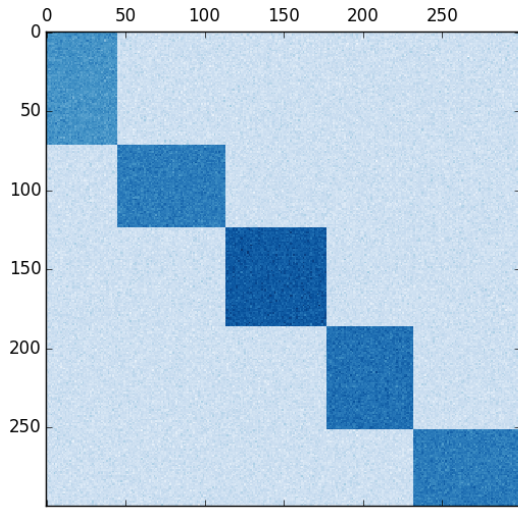


Figure 5.1: Spectral CoClustering Result

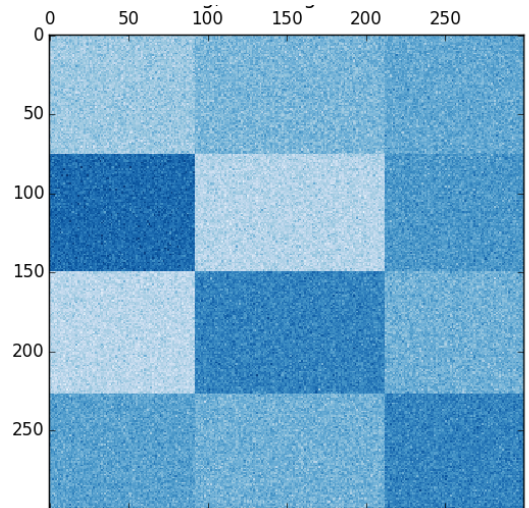


Figure 5.2: Spectral BiClustering Result

Both of these algorithms have advantages and disadvantages. It is said that the majority of gene expression data has an underlying checkerboard structure [17]. This makes sense because of the fact that genes can be active under many different conditions. Even still, we were curious about the results we would get with both methods. These results are discussed in section 5.3.

5.1.2 GUI

Another fantastic tool we were able to use throughout our research was a graphical user interface created by a fellow student, Jeff Borwey. The GUI has the ability to do many tasks in regards to clustering research such as preprocessing of data, graph generation, six clustering algorithms, and internal and external evaluation of resulting clusters. The GUI can be seen in figure 5.3. This figure shows the ‘Preprocessing’ tab, which is what opens up first when running the GUI. This tab allows the user to select a data file to process. The user can choose different normalizations like the geometric or arithmetic mean or can choose no preprocessing at all.

The ‘Cluster’ tab, which can be seen in figure 5.4, is another important tab for our research. The user selects the dataset that they wish to cluster and then chooses which

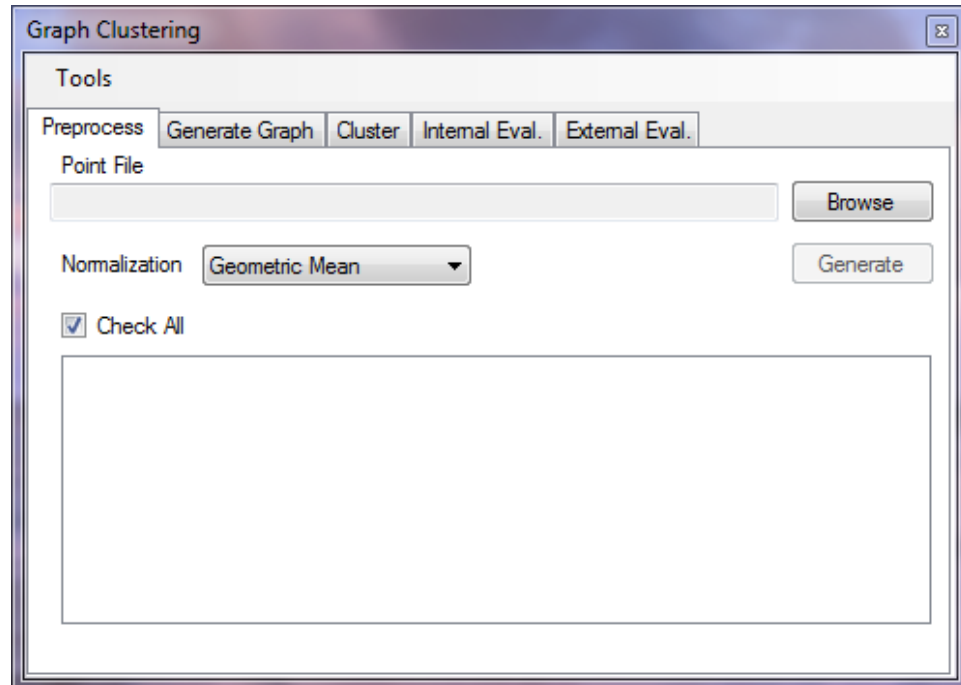


Figure 5.3: Graphical User Interface

clustering algorithm to run on that data. It is very easy to quickly run all of the clustering algorithms on a dataset. The resulting clusters are output to a file named according to the algorithm that was run. All of these files are labeled as *.cluster* files so they are easily accessible.

Finally, the ‘External Evaluation’ tab. The idea of how to evaluate the resulting clusters is discussed in the following section, but the general idea is that we can see how well the algorithm found each of the clusters based on prior knowledge about the dataset. The user can select a *.cluster* file as well as the data file containing the correct clustering and run this evaluator. The accuracy percentage for each cluster and the overall accuracy is output to the screen.

5.2 Result Validation

There are two techniques to determine the accuracy of the results given by a clustering algorithm: internal and external validation. Internal validation is based solely on the

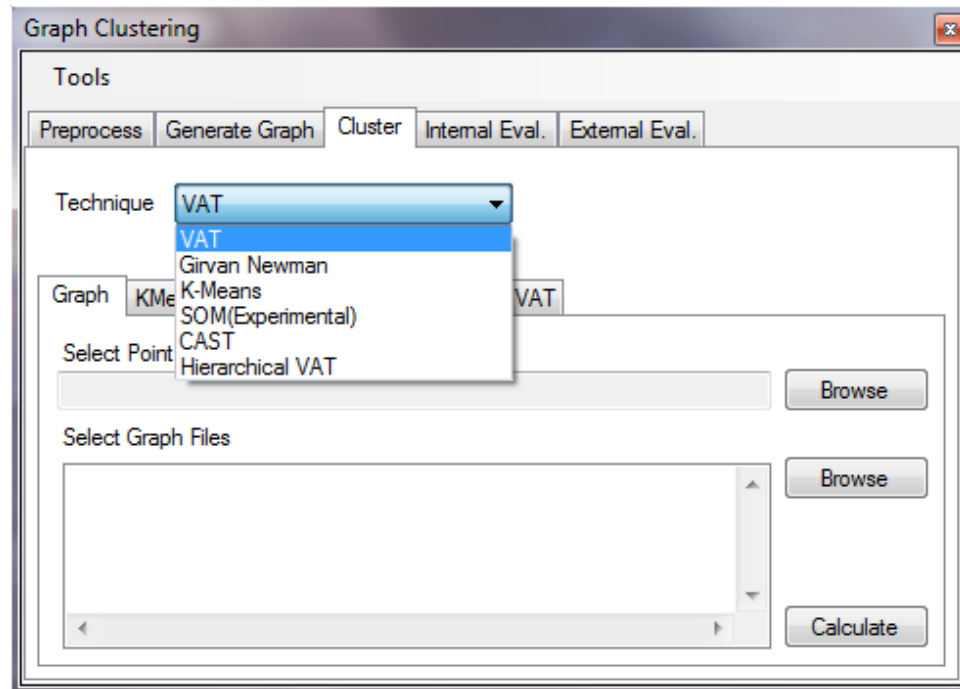
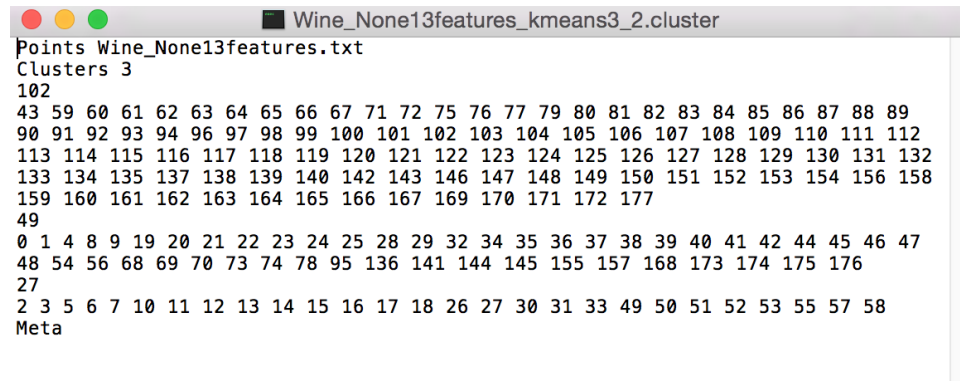


Figure 5.4: Cluster Tab

data and resulting clusters alone. No prior knowledge of the data is necessary to compute internal validation. There are many different indices that can be computed based on the shape of a cluster or how compact a cluster is. Some examples of internal indices are the silhouette index, the Davies-Bouldin index, and the Dunn index [20]. All of these are computed using different equations and tend to favor certain resulting cluster shapes. For example, the silhouette index tends to prefer clusters with more a circular shape [20]. On the other hand, external validation can be used when there is previous knowledge of the data. So, if you have a dataset with already known class or cluster labels then it is easy to just look at your resulting clusters and compare them to the given labels. All of the datasets that we ran experiments on were pre-labeled so we decided to use external validation to calculate the accuracy of the clustering algorithms we chose. Our main goal was to be able to match the pre-defined cluster labels.

5.2.1 Error automation

When we first started our research we would calculate the accuracy or error percentage of the resulting clusters by hand. This became a nuisance. With the amount of algorithms and resulting cluster files, it was necessary to automate the error calculation. All of our resulting clusters were output to different *.cluster* files as mentioned before. These *.cluster* files all have the same format which made it easy to automate the calculation. An example *.cluster* file is shown in figure 5.5. The file name is listed first followed by the number of resulting clusters. The next number represents the number of nodes corresponding to the first resulting cluster. Then the nodes of that resulting cluster are listed. This format continues for the total number of result clusters. Finally, there is a spot for any sort of metadata necessary for the file. Some files, like the one shown, do not have any information in their metadata section.



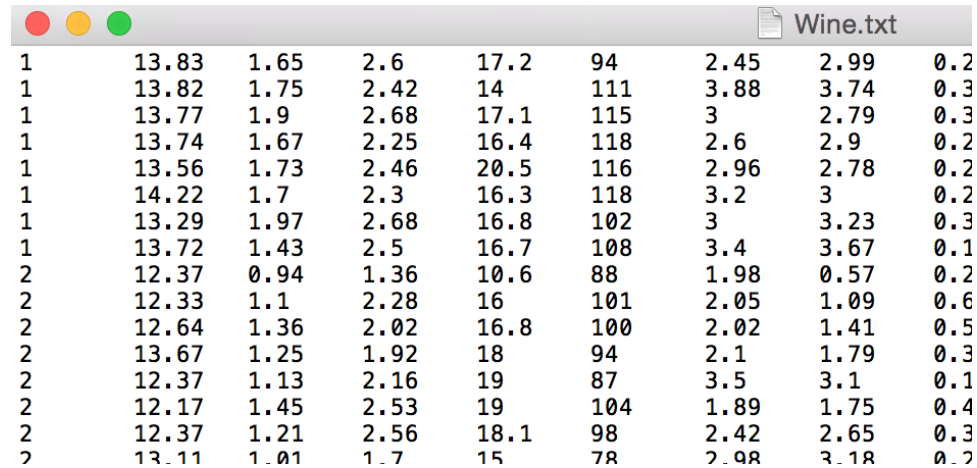
```

Points Wine_None13features.txt
Clusters 3
102
43 59 60 61 62 63 64 65 66 67 71 72 75 76 77 79 80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112
113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
133 134 135 137 138 139 140 142 143 146 147 148 149 150 151 152 153 154 156 158
159 160 161 162 163 164 165 166 167 169 170 171 172 177
49
0 1 4 8 9 19 20 21 22 23 24 25 28 29 32 34 35 36 37 38 39 40 41 42 44 45 46 47
48 54 56 68 69 70 73 74 78 95 136 141 144 145 155 157 168 173 174 175 176
27
2 3 5 6 7 10 11 12 13 14 15 16 17 18 26 27 30 31 33 49 50 51 52 53 55 57 58
Meta

```

Figure 5.5: Example *.cluster* File

With these files, as well as the data files containing the correct clustering in one of the columns (like column one in figure 5.6), we were able to create a count matrix that contains the number of nodes placed in each result cluster that belong to each real cluster. For example, the Wine dataset has three known clusters. The first 59 nodes (0 - 58) belong to cluster one. Nodes 59 through 129 belong to the second cluster. Finally, the rest of the nodes (130 - 177) belong to the third cluster. We consider these the “real” clusters. Now,



1	13.83	1.65	2.6	17.2	94	2.45	2.99	0.2
1	13.82	1.75	2.42	14	111	3.88	3.74	0.3
1	13.77	1.9	2.68	17.1	115	3	2.79	0.3
1	13.74	1.67	2.25	16.4	118	2.6	2.9	0.2
1	13.56	1.73	2.46	20.5	116	2.96	2.78	0.2
1	14.22	1.7	2.3	16.3	118	3.2	3	0.2
1	13.29	1.97	2.68	16.8	102	3	3.23	0.3
1	13.72	1.43	2.5	16.7	108	3.4	3.67	0.1
2	12.37	0.94	1.36	10.6	88	1.98	0.57	0.2
2	12.33	1.1	2.28	16	101	2.05	1.09	0.6
2	12.64	1.36	2.02	16.8	100	2.02	1.41	0.5
2	13.67	1.25	1.92	18	94	2.1	1.79	0.3
2	12.37	1.13	2.16	19	87	3.5	3.1	0.1
2	12.17	1.45	2.53	19	104	1.89	1.75	0.4
2	12.37	1.21	2.56	18.1	98	2.42	2.65	0.3
2	13.11	1.01	1.7	15	78	2.08	3.18	0.2

Figure 5.6: Data File Containing the Correct Clustering

looking at the *.cluster* file, we can see that the first “result” cluster contains one node from the first real cluster, thirty-one nodes from the second real cluster, and twenty-seven nodes from the third real cluster. We can add up these counts and place them in the count matrix like table 5.1. Based on these counts, we can assign each result cluster to a real cluster and calculate the error percentages. This can be done in multiple ways.

The first technique we decided to try was a greedy approach. This approach simply finds the largest number in the count matrix, assigns the result cluster to the corresponding real cluster, and then marks the row and column as used by setting all of the cells to a negative number. This process continues until each result cluster has been assigned to a real cluster. Going back to the wine example, we can see that the greedy approach would first chose to assign result cluster two to real cluster one since the largest number in the matrix is 64. Next, result cluster one would be assigned to real cluster two. Finally, result cluster three would be assigned to real cluster three with a total of 95 nodes correctly assigned. It is somewhat intuitive that this does not always give the best results. In this example alone we can see that by assigning result cluster three to real cluster three we do not add any more nodes to our overall accuracy (since they have zero in common). Although this is true, the greedy approach finds the optimal solution the majority of the

	Real One	Real Two	Real Three
Result One	1	31	27
Result Two	64	7	0
Result Three	37	11	0

Table 5.1: Wine K -means Results

time and runs extremely fast.

We determined that it would be beneficial to also have the option of finding the optimal solution every time as well. So, we implemented the brute force solution as our second technique to the error calculation. To accomplish this, we take in to consideration the number of real and resulting clusters. By taking the permutation of all resulting cluster numbers we can find every possible option for a solution. If there are three result clusters then we would find the following permutations:

$$(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), \text{ and } (3,2,1)$$

We consider the first number to be assigned to the first real cluster, the second number to the second real cluster, and so on. This gives us all possible combinations of assigning resulting clusters to real clusters. If there are more result clusters than real clusters, we only consider the first so many numbers. For example, if there are five result clusters and only three real clusters then we would only consider the first three numbers in every permutation. This still accurately gives us all possible solutions. We can do the same for the opposite situation with more real clusters than resulting clusters. Everything is just completed vice versa. While going through all of the permutations, we keep track of the current maximum accuracy. If we find a permutation that gives better accuracy, we save it and continue on until all possible options have been tried. This gives us the best possible pairings. Going back to the Wine dataset and table 5.1, we get the following counts and totals corresponding to the permutations above:

$$(1+7+0=8), (1+0+11=12), (31+64+0=95),$$

$$(31+0+37=68), (27+64+11=102), (27+7+37=71)$$

The fifth permutation (3,1,2) gives us the best result with 102 nodes correctly assigned. Although this percentage is not vastly bigger than the greedy approach, we still get better overall accuracy. For our implementation, the brute force algorithm will run as long as there is less than ten real clusters and less than ten result clusters. Seeing how the majority of our datasets have a small number of clusters, this is not an issue. It is also important to note that there exists a polynomial time solution to this problem called the Hungarian method or the Kuhn-Munkres algorithm [18]. Essentially the algorithm manipulates the count matrix in different ways until a solution is found. This method would of course be the most efficient method to use, but our brute force algorithm achieves the correct result and does not take very long to compute the solution. So, we chose not to implement the Hungarian method.

5.3 Results

The goal of this research was to dive into clustering, try different combinations of preprocessing, parameters, and algorithms and see what kind of results we could get. As we went through our research, we continuously attempted to get better results by trying new techniques. The timeline of these experiments can be seen throughout this chapter. Also, it should be acknowledged that all of the percentages in this section were calculated using the brute force approach described in the prior section.

Before jumping into the accuracy of our experiments, it is necessary to understand what each of the listed algorithms actually stands for. The given algorithm names are parallel to those listed in Chapter 3. Also, we constructed two different types of graphs to test these algorithms on. The first type of graph is the Geometric threshold graph. This graph only contains edges whose length is within some threshold r . For our experiments, we calculate the minimum spanning tree (MST) of the graph and consider r to be the length of the longest edge in the MST. This gives us one version of a minimum connectivity

graph. The second type of graph we use is the k -nearest neighbors graph. This graph only allows each node to have k neighbors. For our implementation, we calculate this k -value by finding the minimum number of neighbors each node must have in order for the graph to stay connected. These are represented as **GEO** and **KNN** respectively. Also, each time VAT-Clust runs, it can be run on a weighted or an unweighted graph. These are labeled appropriately as ‘with weights’ and ‘w/o weights’. Finally, Hierarchical VAT-Clust can determine how to recurse through the clusters by using two different options. The first, smallest VAT, calculates the VAT value for each of the clusters of the given results and recursively runs VAT-Clust on the cluster with the smallest VAT value. Similarly, the second option will recursively run VAT-Clust on the cluster with the largest number of nodes. This is considered the ‘largest remaining’ option.

5.3.1 *Clustering the real datasets*

The first set of experiments we ran were over the real datasets mentioned in section 4.3.3. We wanted to see how well our chosen algorithms would run on data that had a well known, already labeled structure. The accuracy percentages can be seen in table 5.2. Looking at the overall average for each dataset, we can see that the Iris dataset was the most successful clustering overall at 74.49% accuracy with the Breast Wisconsin dataset coming in as a close second at 73.68% accuracy. Looking into the Iris dataset in more detail, we can see that the Girvan Newman on a KNN graph had the best accuracy at 96.67% . The Hierarchical VAT-Clust on a KNN graph without weights had 96% accuracy regardless of the recursive option used. A visualization of this result can be seen in Figure 5.7. In part 1 of Figure 5.7 we can see the first attack on the dataset. VAT splits the data into two clusters. The smaller of the two contains 50 nodes that all belong to the real first cluster. The second contains the rest of the 100 nodes. In part 2 of Figure 5.7 we can see the larger 100 node cluster recreated into a new KNN graph. Finally, in part 3 we can see the final attack that splits the 100 node cluster into two clusters with 46

		Breast	Breast WI	Ecoli	Iris	Wine
Girvan Newman GEO (k)		21.30	63.44	45.24	67.33	60.67
Girvan Newman KNN (k)		50.94	76.80	60.71	96.67	71.35
K-means (k)		30.75	85.41	58.15	89.11	57.30
CAST		19.81	72.06	65.18	69.33	48.88
VAT GEO	with weights	20.75	N/A	44.05	66.67	64.61
	w/o weights	20.75	N/A	44.05	66.67	64.61
VAT KNN	with weights	34.91	64.67	62.80	66.67	64.61
	w/o weights	34.91	81.20	62.80	66.67	67.42
Hier-VAT GEO	Smallest VAT	33.96	-	43.75	78.67	71.35
	Small VAT w/o weights	33.96	-	64.58	72.00	71.35
	Largest Remaining	34.91	-	44.05	78.67	71.35
	LR w/o weights	38.68	-	76.79	72.00	71.35
Hier-VAT KNN	Smallest VAT	36.79	-	47.62	51.33	66.85
	Small VAT w/o weights	36.79	-	76.49	96.00	72.47
	Largest Remaining	43.40	-	55.65	89.33	69.10
	LR w/o weights	43.40	-	53.57	96.00	72.47
Average		32.37	73.68	55.96	74.49	66.35

Table 5.2: Real Data Results

and 54 nodes. Hierarchical VAT-Clust incorrectly clusters only four nodes and Girvan Newman incorrectly clusters only three nodes. These are fantastic results.

Overall, we found that the algorithms we chose can cluster this biological data fairly well. Some datasets, such as the Breast dataset, are slightly harder to cluster. In fact, looking at the table above as well as table 5.8, we can see that the Breast dataset had the worst percentages out of all the datasets. We believe that this dataset would have much better results from using supervised clustering methods rather than the purely unsupervised clustering methods we use. Regardless of the Breast dataset results, we thought these were good algorithms to continue on to use for clustering the gene expression data.

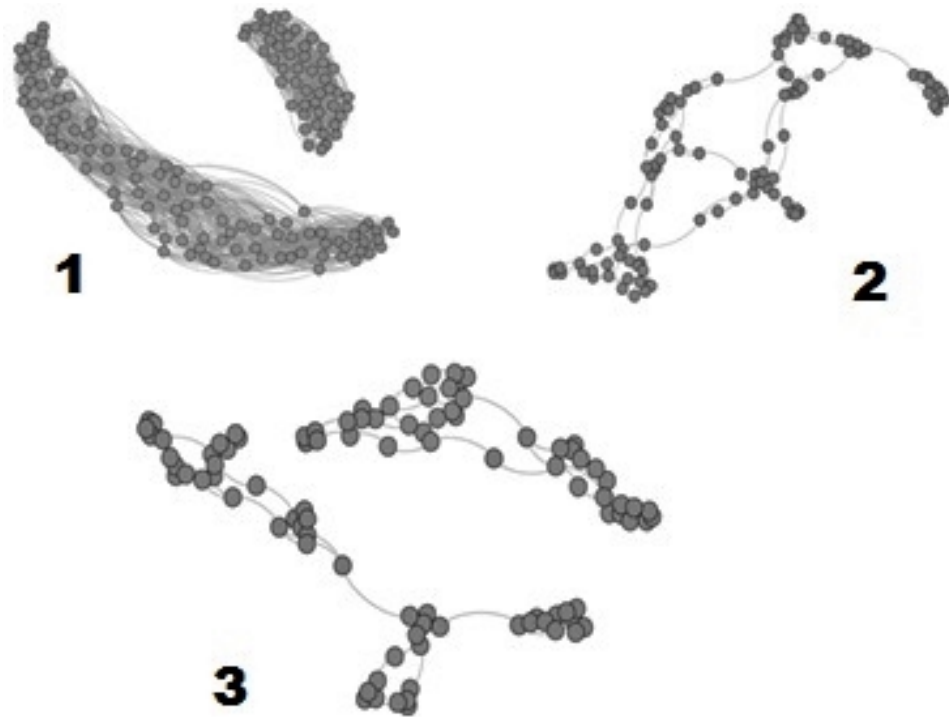


Figure 5.7: Hierarchical VAT-Clust Visualization of Iris Dataset

5.3.2 Clustering the post-mortem data using all genes

The next set of experiments we ran were over the post-mortem gene expression data using all of the genes in the given dataset. The first run we executed consisted of the entire dataset. We attempted to break the data up into the four known sample clusters. Our overall average accuracy percentage for this data was 33.04%. Clearly, this is not what we were wanting. We can see that the Girvan Newman on a KNN graph was again one of the top clusterings. One thing different about this data was that the Hierarchical VAT-Clust on a GEO graph actually did better than on a KNN graph which was a little surprising because the KNN graph was almost always better than the GEO graph on the real datasets. The reason why our results were so poor is that the average degree of the 50 nodes is 41.64 with a total edge count of 1,041. This minimum connectivity graph can be seen in figure 5.8. Just by looking at the graph it is clear that this data would be difficult to cluster.

	AllSamples	CvBP	CvS
Girvan Newman GEO (k)	34.00	61.54	57.14
Girvan Newman KNN (k)	36.00	50.00	60.71
K-means (k)	34.80	62.69	60.36
CAST	32.00	57.69	53.57
VAT GEO with weights	32.00	61.54	N/A
w/o weights	32.00	61.54	N/A
VAT KNN with weights	32.00	53.85	57.14
w/o weights	32.00	53.85	57.14
Hier-VAT GEO Smallest VAT	34.00	-	-
Small VAT w/o weights	36.00	-	-
Largest Remaining	34.00	-	-
LR w/o weights	36.00	-	-
Hier-VAT KNN Smallest VAT	32.00	-	-
Small VAT w/o weights	32.00	-	-
Largest Remaining	32.00	-	-
LR w/o weights	32.00	-	-
Average	33.04	57.79	57.54

Table 5.3: Results Using All Genes

The next two runs we executed still used all of the genes, but instead of looking for the four different sample groups we only look to split the control group from the bipolar samples, and the control group from the schizophrenia samples. It should be noted that when running experiments that only bipartition the graph (i.e. there are only two clusters) it is pointless to run Hierarchical VAT-Clust. Since we give Hierarchical VAT-Clust the number two as the minimum number of clusters, it will run equivalently to regular VAT-Clust without any recursive attacks. That is why there are no results shown in table 5.3 for this clustering algorithm. This can also be seen in some of the other tables as well (5.4 & 5.5).

The average percentages for *CvBP* and *CvS* are both better than the *AllSamples* sitting above 57%. Looking at how each of the algorithms performed, we can see that the *k*-means algorithm was the best for splitting the control samples and the bipolar

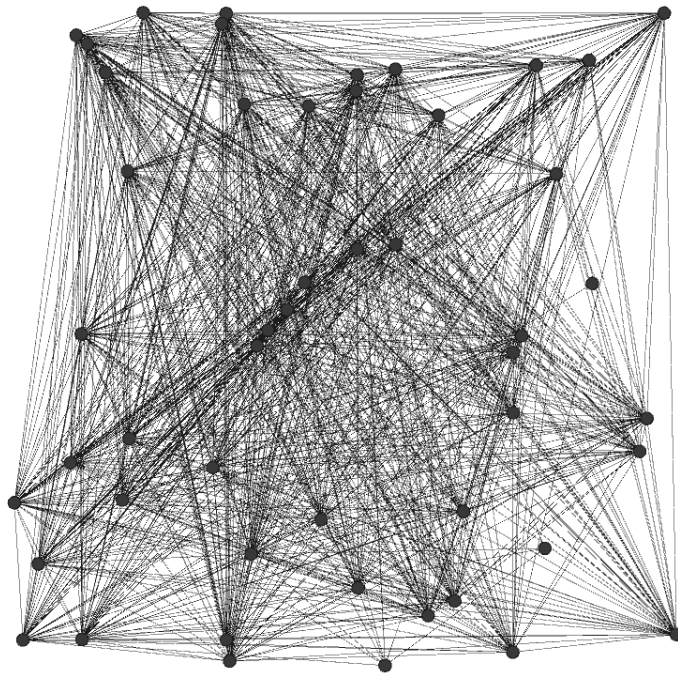


Figure 5.8: GEO Graph of *AllSamplesAllGenes* Dataset

samples at 62.69%. Also, the Girvan Newman and VAT-Clust algorithms on a GEO graph performed fairly well at 61.54%. As for splitting the control samples and the schizophrenia samples, the Girvan Newman algorithm run on a KNN graph performed the best at 60.71% with the k -means algorithm right behind that at 60.36%. We were unable to gather results for VAT-Clust on the GEO graph because the algorithm would not converge. Although these results are better than the 33.04% from before, they were still not the results we were hoping for.

5.3.3 Using differentially expressed genes

We figured that the reason our results were poor was simply due to the fact that there is a large amount of data associated with this set. We needed some way of finding the informative genes, or the genes that actually help find the clusters within the data. This is where the differentially expressed genes come in to play. Since we were able to find differentially expressed genes with respect to bipolar disorder from [13] and with respect to

		AllvsBP	CvBP	BiPart
Girvan Newman GEO (k)		40.00	65.38	70.00
Girvan Newman KNN (k)		40.00	53.85	58.00
K-means (k)		36.67	62.69	72.00
CAST		38.00	57.69	70.00
VAT GEO	with weights	38.00	65.38	70.00
	w/o weights	38.00	65.38	70.00
VAT KNN	with weights	34.00	53.85	58.00
	w/o weights	34.00	53.85	58.00
Hier-VAT GEO	Smallest VAT	36.00	-	-
	Small VAT w/o weights	40.00	-	-
	Largest Remaining	38.00	-	-
	LR w/o weights	40.00	-	-
Hier-VAT KNN	Smallest VAT	40.00	-	-
	Small VAT w/o weights	40.00	-	-
	Largest Remaining	40.00	-	-
	LR w/o weights	40.00	-	-
Average		37.83	59.71	65.17

Table 5.4: Results Using Bipolar Genes

schizophrenia from [7], we assumed that this would help differentiate the hidden clusters and give us better results. We make this hypothesis based on the method in which these genes are found. Differential expression is obtained with respect to the average level of expression across the control group. To put it in (somewhat) simple steps, the average expression level of each gene across all of the control samples is calculated. Then we divide each of the non-control samples genes by the average calculated above to acquire a ratio called the “fold change”. Then the logarithm of this ratio is calculated so that it becomes the “log fold change”. Finally, if the log fold change is over some threshold value then the gene is considered differentially expressed. From here on, we refer to these genes in our experiments as the bipolar genes and the schizophrenia genes. The GEO graphs of these two subsets of data can be seen in figures 5.9 and 5.10.

The accuracy percentages obtained from using only bipolar genes can be viewed in

table 5.4. From this table we can see that our overall average accuracy percentages were slightly better. Using the bipolar genes, we were able to correctly cluster a little over 4% more than before when attempting to find all four of the sample groups. The majority of the algorithms produced similar accuracy results with the highest and most common percentage being 40% and the lowest percentage being 34%. On top of this, we tried to once again find just the control samples and the bipolar samples (figure 5.9). It is interesting to note that both the Girvan Newman and VAT-Clust on the GEO graph correctly cluster 65.38% of the nodes. This percentage was much higher than what we were used to so this made us a little happier. Finally, we attempted to find the bipolar group out of all of the samples. To do this, we created a new dataset that labeled the control samples, schizophrenia samples, and major depression samples as one group and labeled the bipolar samples as a separate group. We expected that using only the bipolar genes would allow the algorithms to separate the bipolar samples from the rest of the samples. These results can be seen in table 5.4 as well. This group gave us the highest overall average accuracy at 65.17% with the k -means algorithm averaging out to our highest percentage of 72%. These results were definitely more of what we were hoping for.

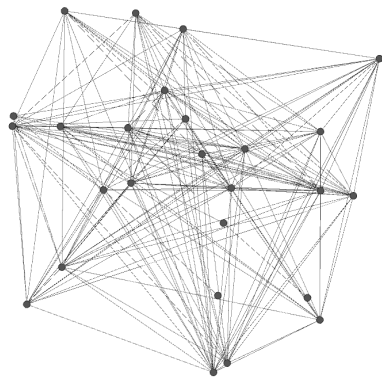


Figure 5.9: GEO Graph of *CvBP*

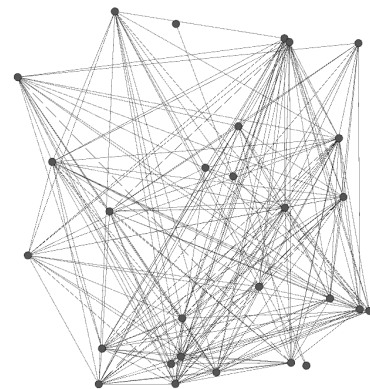


Figure 5.10: GEO Graph of *CvS*

We ran the same three experiments using the schizophrenia genes instead of the

		AllvsSch	CvS	BiPart
Girvan Newman GEO (k)		36.00	57.14	72.00
Girvan Newman KNN (k)		40.00	64.29	66.00
K-means (k)		36.67	56.07	66.40
CAST		38.00	46.43	72.00
VAT GEO	with weights	32.00	N/A	72.00
	w/o weights	32.00	N/A	72.00
VAT KNN	with weights	34.00	64.29	62.00
	w/o weights	42.00	64.29	68.00
Hier-VAT GEO	Smallest VAT	32.00	-	-
	Small VAT w/o weights	32.00	-	-
	Largest Remaining	32.00	-	-
	LR w/o weights	32.00	-	-
Hier-VAT KNN	Smallest VAT	38.00	-	-
	Small VAT w/o weights	42.00	-	-
	Largest Remaining	38.00	-	-
	LR w/o weights	42.00	-	-
Average		35.53	60.13	68.70

Table 5.5: Results Using Schizophrenia Genes

bipolar genes. These results can be seen in table 5.5. We can see that these genes gave our algorithms roughly the same accuracy values. Attempting to cluster the data into the four sample groups was a little worse with the schizophrenia genes, but bipartitioning the data into Control versus Schizophrenia (*CvS*) and Schizophrenia versus everything else (*BiPart*) both actually did a little better than the respective bipolar experiments.

Table 5.6 shows a summary of each algorithm’s accuracy percentage when attempting to find all of the sample groups. So, the first column comes from the entire dataset (All Samples and All Genes), the second column comes from the subset in which we try to find all of the samples using only bipolar genes, and the third is similar to the second but with the schizophrenia genes instead. This table shows us that although our results did not greatly improve by using the differentially expressed genes, they still improved. We were able to get +4% accuracy using the bipolar genes and +2% using the schizophrenia

	AllSamples	AllvsBP	AllvsSch
Girvan Newman GEO (k)	34.00	40.00	36.00
Girvan Newman KNN (k)	36.00	40.00	40.00
K-means (k)	34.80	36.67	36.67
CAST	32.00	38.00	38.00
VAT GEO with weights	32.00	38.00	32.00
w/o weights	32.00	38.00	32.00
VAT KNN with weights	32.00	34.00	34.00
w/o weights	32.00	34.00	42.00
Hier-VAT GEO Smallest VAT	34.00	36.00	32.00
Small VAT w/o weights	36.00	40.00	32.00
Largest Remaining	34.00	38.00	32.00
LR w/o weights	36.00	40.00	32.00
Hier-VAT KNN Smallest VAT	32.00	40.00	38.00
Small VAT w/o weights	32.00	40.00	42.00
Largest Remaining	32.00	40.00	38.00
LR w/o weights	32.00	40.00	42.00
Average	33.04	37.83	35.53

Table 5.6: Comparison with All Samples

genes.

Table 5.7 shows a summary of each algorithm’s accuracy percentage when attempting to separate the control group from the bipolar samples and from the schizophrenia samples. The first two columns represent the accuracies when trying to separate the control group from the bipolar samples using all of the genes and using only the bipolar genes. The third and fourth columns are similar but with the schizophrenia samples instead. Like the situation above, we were still able to improve our accuracy percentages by using the differentially expressed genes. When splitting the control and bipolar samples or schizophrenia samples, we were able to get +2% and +3% accuracies respectively using the differentially expressed genes.

	CvsBP All	CvsBP BP	CvS All	CvS Sch
Girvan Newman GEO (k)	61.54	65.38	57.14	57.14
Girvan Newman KNN (k)	50.00	53.85	60.71	64.29
K-means (k)	62.29	62.69	60.36	56.07
CAST	57.69	57.69	53.57	46.43
VAT GEO with weights	61.54	65.38	N/A	N/A
w/o weights	61.54	65.38	N/A	N/A
VAT KNN with weights	53.85	53.85	57.14	64.29
w/o weights	53.85	53.85	57.14	64.29
Average	57.79	59.71	57.54	60.13

Table 5.7: Control vs. Other Samples

5.3.4 Looking into the details of the results

Since the results were better when using the differentially expressed genes, we decided to look into this a little further and were able to see some interesting details. One thing we noticed is that our results being better was not necessarily parallel to actually separating particular samples from the others. For example, take a look at figure 5.11. From this *.cluster* file we can analyze in detail how the clustering of the Girvan Newman on a GEO graph ran. The bipolar samples are labeled by numbers 26 through 36 and all of the other samples are everything else. The result of this particular file had an overall accuracy of 70%, but looking at the file we can see that only one of the bipolar samples was actually pulled away from the large cluster (node 36). The rest of the bipolar samples stayed in the same cluster as everything else. The accuracy of the first cluster ended up being 87.18% which makes sense because only a few of the samples were incorrectly pulled out of the big set. The cluster we were actually interested in only had 9.09% accuracy (1 sample correct out of 11 total). This is not what we were expecting at all. If we look at figure 5.12, we can see that there is some overlap between the differentially expressed genes for samples with major mental disorders [13]. Even though we were attempting to pull out the bipolar samples from all of the other samples using what we know to be differentially

expressed genes with respect to bipolar disorder, it is possible that some of these genes are also differentially expressed with respect to the other major mental disorders as well. Therefore, it would most definitely be difficult to pull the bipolar samples out using only these genes.

```

BiClust_BPGenes_None90features_Euclidean_732_GN2.cluster
Points BiClust_BPGenes_None90features.txt
Clusters 2
44
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 21 22 23 24 25 26 27 28 29 30 31 32 33
34 35 37 39 40 43 44 46 47 48 49
6
19 36 38 41 42 45
Meta girvan_newman

```

Figure 5.11: BiPart_BPGenes .cluster File

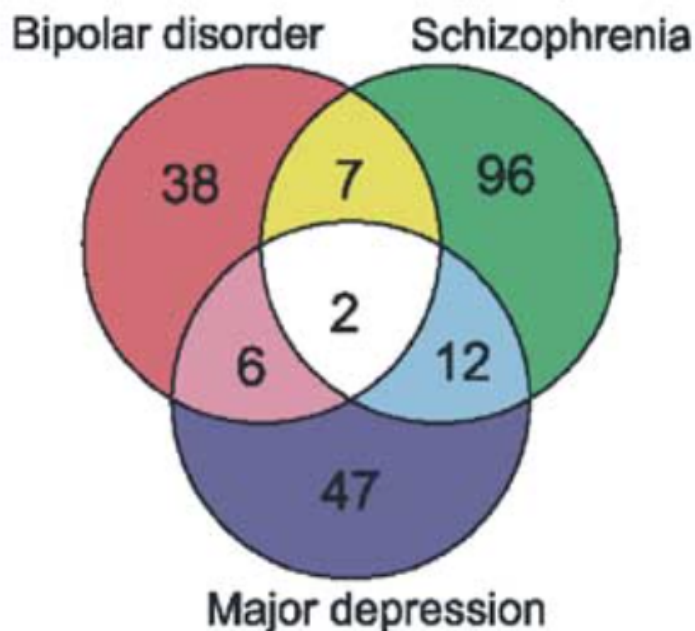


Figure 5.12: Venn Diagram Showing Differential Gene Numbers

On the other end of the spectrum, our clustering algorithms did much better when attempting to partition the control samples and bipolar samples or the control samples and schizophrenia samples. Take a look at figures 5.13 and 5.14. These are resulting

.cluster files showing VAT-Clust’s attempt at the two separations described above. In these examples, the control samples are labeled by numbers 0 through 14 while the labels 15 through 25 (or 27) are for the bipolar (or schizophrenia) samples. The interesting thing to note here is that VAT does attempt to separate these as it should. For the control versus bipolar example (figure 5.13), two of the correct bipolar nodes were pulled apart as a second cluster (nodes 15 and 25). Also, under the “Meta Vat” we can see that nodes 22 and 23 were the two nodes removed as the critical node set. This implies that VAT-Clust considers them to be important nodes. Unfortunately with the greedy approach, these nodes were added back in to the first cluster instead of being added to their appropriate cluster.

```

CvsBP_BPGenes_None90features_Euclidean_186_VAT.cluster
Points CvsBP_BPGenes_None90features.txt
Clusters 2
24
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 16 17 18 19 20 21 22 23 24
2
15 25
Meta VAT 0.666667
22 23

```

Figure 5.13: CvBP *.cluster* file

```

CvS_SchGenes_None135features_Euclidean_KNN_2_VAT.cluster
Points CvS_SchGenes_None135features.txt
Clusters 2
9
0 1 4 16 21 22 24 25 27
19
2 3 5 6 7 8 9 10 11 12 13 14 15 17 18 19 20 23 26
Meta VAT 0.1
26

```

Figure 5.14: CvS *.cluster* file

We can see that VAT-Clust similarly attempts to pull out the schizophrenia samples from the control samples in figure 5.14. The first cluster has six of the correct schizophrenia nodes but also has three incorrect nodes. Similar to the bipolar example, the node in the “Meta VAT” section is one of the nodes we were looking to separate out but also got

added back in to the incorrect cluster. Although they were not completely partitioned as they should be, it is interesting to see that VAT-Clust at least began to partition these datasets the way they should be. These are promising results that could potentially lead to even better accuracy in the future.

5.3.5 *An attempt at biclustering*

Biclustering was a popular topic throughout the research we did on clustering gene expression data. As described in section 5.1.1, there were two biclustering algorithms already implemented in the *scikit-learn* module. Using these, we were able to bicluster our data. These results can be viewed in table 5.8. Since we still did not know an actual clustering for the genes, we simply calculate these accuracies based solely on how well the samples were clustered. The Iris dataset was easily partitioned by the Spectral BiClustering algorithm with an accuracy of 98%. Also, the Breast Wisconsin dataset had good results for both the Spectral BiClustering and Co-Clustering algorithms with 85% and 87% respectively. Unfortunately, these algorithms did not really help us to get better results with our gene expression data. There are very few percentages that are better than the ones discussed before, and even when they are better there is not a significant difference. The one conclusion we can draw from these results is that six out of the nine different gene expression datasets had better results using the Spectral Biclustering algorithm. This makes sense because this algorithm allows the data to belong to multiple biclusters, whereas the Spectral Co-Clustering algorithm allows the data to belong to only one bicluster. Since genes are often expressed during multiple scenarios, we expected them to possibly belong to multiple biclusters.

5.3.6 *Discussion of results and contributions*

Overall, we were quite surprised with how poorly CAST performed since the algorithm was created specifically for gene expression data. It seems as though it was once known

Dataset	Spec BiClust	Spec CoClust
Wine	16.85	23.60
Iris	98.00	10.00
Breast	0.943	23.58
Breast_WI	85.24	87.35
Ecoli	38.69	0.00
AllGenesAllSamples	34.00	16.00
CvBP_AllGenes	30.77	46.15
CvBP_BPGenes	30.77	0.00
AllvsBP_BPGenes	26.00	30.00
BiPart_BPGenes	34.00	64.00
CvS_AllGenes	39.29	10.71
CvS_SchGenes	42.86	28.57
AllvSch_SchGenes	22.00	20.00
BiPart_SchGenes	56.00	42.00

Table 5.8: Biclustering Results

as a great algorithm, but has since been proven to be outdated. Except for CAST, each of the algorithms we ran experiments with had the highest accuracy at some point. This is important because it shows that no one algorithm is particularly better than another with these types of datasets.

When looking into our new algorithm, we can see that running VAT-Clust and Hierarchical VAT-Clust without weights does slightly better overall than with weights. Hierarchical VAT always outperforms the original VAT-Clust algorithm. On top of that, using a GEO graph versus using a KNN graph made a little difference. Using the GEO graph on the gene expression data resulted in better accuracies, while using the KNN graph on the real datasets resulted in better accuracies. Finally, with these preliminary results, it seems as though using the smallest VAT tended to give slightly better results than recursively clustering over the largest remaining cluster.

The last thing we would like to point out is that we attempted running these experiments by first calculating the logarithmic transformation on the data matrix. We also calculated the logarithmic transformation of the distance/similarity matrix to see if it

would make a difference. Since much of the prior research said that this was a good technique to use on gene expression data, we thought it would produce better results. Sadly, we received the exact same results by doing these preprocessing techniques. Once again, we are not quite sure why this did not work as we had expected. The one thing we speculate is that the data had already been preprocessed, and attempting to preprocess this data again did not have any effect on the resulting clusters.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

We wanted to examine and investigate different clustering approaches for biomedical datasets and answer the questions listed in Chapter 1. We note that hierarchical clustering is a popular technique, especially for gene expression data, but stick with the graph-theoretic approach that has recently gained more interest. We choose to explore five different clustering algorithms. The very popular k -means algorithm clusters data into k clusters by minimizing the squared error distortion of the given clusters. The Cluster Affinity Search Technique is a practical heuristic that was intended to be used as a clustering agent on gene expression data. It assigns nodes to clusters by determining if the given node is “close” to a cluster (i.e. if it is within a distance threshold of the cluster). The third and fourth algorithms are very similar and use a new technique to determine clusters. VAT-Clust uses the resilience of a graph and a node-based removal to create disconnected components that become clusters. Similarly, Hierarchical VAT-Clust recursively runs VAT-Clust on resulting clusters until a minimum number of clusters provided as a parameter has been reached. Finally, the Girvan-Newman algorithm focuses on the betweenness of a graph. The edges with the highest betweenness values can be removed from the graph, eventually creating separate, connected components that will be considered the resulting clusters.

Using these clustering algorithms, we attempt to cluster many different datasets. The main dataset we wished to explore was a post-mortem gene expression dataset containing gene expression profiles of brains with major mental disorders such as bipolar disorder, major depression, and schizophrenia. We also created subsets of this data based on the knowledge that differentially expressed genes can greatly help identify certain classifications. Lastly, we tested our chosen clustering algorithms on five real datasets

(Breast, Breast Wisconsin, Ecoli, Iris, and Wine) pulled from the UCI Machine Learning Repository. These datasets have already labeled, well-known clusters so it is easy to test the accuracy of the resulting clusters.

Jeff Borwey, a fellow student, helped tie all of this together by creating a graphical user interface (GUI) that implemented all of the algorithms, preprocessing, graph creation, and cluster validation in one central location. This made it so much easier to test many different parameters for each algorithm, as well as different graph constructions of the data. The version we used for our research was the first committed version on March 15, 2015 [4]. On top of Jeff's GUI, we also used scikit-learn, a python module with many different tools including the biclustering submodule that we were able to use. Once all of these experiments were finished running, we needed some way to calculate the accuracy of the results.

We implemented two different error/accuracy percentage calculators. The first, a greedy approach, simply finds the largest number of correct nodes in a cluster and assigns that resulting cluster to the appropriate real cluster. This process continues until all resulting clusters have been assigned to a real cluster. The second approach we implemented was the brute force approach. This finds all of the possible combinations of real and resulting clusters, and picks the pairings with the absolute best accuracy percentage. We use the brute force approach for all of our result percentages in this research.

From our results, we can confirm first and foremost that gene expression data is extremely difficult to deal with. Our chosen clustering algorithms did poorly when attempting to cluster the samples using all of the given genes. We then narrowed down the amount of genes we used to cluster the samples by only including genes that were known to be differentially expressed with respect to their given classification (like bipolar genes or schizophrenia genes). We hypothesized that by using only these genes the

clustering algorithms would at least be able to identify the respective class or sample label associated with the genes. Unfortunately, our results did not match up with this theory as well as we hoped. We did see a slight improvement in the accuracies when using the differentially expressed genes, but expected the jump in percentages to be much larger. Finally, we attempted the popular biclustering technique mentioned in much of the gene expression clustering research. The results of the biclustering methods did not improve the accuracies as much as we had expected and were somewhat disappointing.

As for future work, we think it would be smart to attempt to limit the data in a different way. Principal Component Analysis (PCA) seems to be a very popular feature extraction technique that transforms the given data into a smaller, simplified version of itself. By doing this it would be easier to cluster the samples because there would be much less garbage to try to sort through. We think that having a better and more detailed understanding of gene expression data might help to determine why certain approaches work better than others and maybe even give us a better sense of the direction we could take to get better accuracy. Finally, it would be beneficial to attempt these clustering techniques on different sets of gene expression data. There is a possibility that the post-mortem data we chose is just extremely difficult to work with and maybe other datasets would give us a better outcome. In fact, at the very end of our research, we received some information that lead us to believe this is exactly the issue we were running in to.

Microarray technology was a fantastic step in the right direction for gene expression analysis. Having said that, RNAseq technology has become the main focus of current biological research. RNAseq has the same purpose of finding genes that are active during different processes and at different times. However, there are a few differences between RNAseq and microarray technologies. One difference is RNAseq technology does not require the researcher to know or input specific genes to keep an eye on, whereas

microarray technology requires this. So, RNAseq essentially looks at the entire genome and pulls out the important genes for you. Clearly, for biology researchers this is extremely important. For the computer science world and our research in particular, the most important difference between the two is that RNAseq data has been proven to be much more accurate than the microarray data. Why not use RNAseq data instead then? Well, the issue here is that microarray technology is inexpensive compared to RNAseq technology meaning that microarray data is much more publicly available and very easy to find. Also, RNAseq data is super bulky data; even more so than microarray data. So, for people to get this data out on the internet it would be pertinent for them to upload some sort of cleaned up version. This proposes another issue. There are multiple ways of cleaning up data, and the resulting datasets would not necessarily be consistent with one another. To make sure we were getting the data we wanted, we would essentially have to work directly with a biologist in this field with access to RNAseq technology. It would be very nice to have access to this type of data in the future and re-run our experiments to see what kind of results we would get.

REFERENCES

- [1] David B Allison, Grier P Page, T Mark Beasley, and Jode W Edwards. *DNA microarrays and related genomics techniques: design, analysis, and interpretation of experiments*. CRC Press, 2014.
- [2] Michael R Anderberg. Cluster analysis for applications. Technical report, DTIC Document, 1973.
- [3] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297, 1999.
- [4] Jeffrey Borwey. Jeff borwey’s graph clustering github. <https://github.com/JeffBorwey/GraphClustering>, March 2015.
- [5] Jeffrey Borwey, Darla Ahlert, Tayo Obafemi-Ajayi, and Gunes Ercal. A graph-theoretic clustering methodology based on vertex-attack tolerance. In *The Twenty-Eighth International Flairs Conference*, 2015.
- [6] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.
- [7] AL Collins, Y Kim, RJ Bloom, SN Kelada, P Sethupathy, and PF Sullivan. Transcriptional targets of the schizophrenia risk gene mir137. *Translational psychiatry*, 4(7):e404, 2014.
- [8] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.
- [9] Gunes Ercal and John Matta. Resilience notions for scale-free networks. In *Complex Adaptive Systems*, pages 510–515, 2013.
- [10] Kemal Eren, Mehmet Deveci, Onur Küçüktunç, and Ümit V Çatalyürek. A comparative analysis of biclustering algorithms for gene expression data. *Briefings in bioinformatics*, 14(3):279–292, 2013.
- [11] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.
- [12] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.
- [13] K Iwamoto, C Kakiuchi, M Bundo, K Ikeda, and T Kato. Molecular characterization of bipolar disorder by comparing gene expression profiles of postmortem brains of major mental disorders. *Molecular psychiatry*, 9(4):406–416, 2004.

- [14] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: A survey. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 16(11), 2004.
- [15] Neil C Jones and Pavel Pevzner. *An introduction to bioinformatics algorithms*. MIT press, 2004.
- [16] Grainne Kerr, Heather J Ruskin, Martin Crane, and Padraig Doolan. Techniques for clustering gene expression data. *Computers in biology and medicine*, 38(3):283–293, 2008.
- [17] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.
- [18] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- [19] M. Lichman. UCI machine learning repository, 2013.
- [20] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. Understanding of internal clustering validation measures. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 911–916. IEEE, 2010.
- [21] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(1):24–45, 2004.
- [22] Markus Maier, Ulrike V Luxburg, and Matthias Hein. Influence of graph construction on graph-based clustering measures. In *Advances in neural information processing systems*, pages 1025–1032, 2008.
- [23] John Matta, Jeffrey Borwey, and Gunes Ercal. Comparative resilience notions and vertex attack tolerance of scale-free networks. *CoRR*, abs/1404.0103, 2014.
- [24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [25] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.
- [26] Chun Tang, Aidong Zhang, and Jian Pei. Mining phenotypes and informative genes from gene expression data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 655–660. ACM, 2003.

- [27] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.