COMPARING CLUSTERING ALGORITHMS FOR USE WITH GENOMIC AND PROTEOMIC DATA

THESIS

Rebecca Ann Olson, Civilian, USAF

AFIT/GAM/ENC/02S-1

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GAM/ENC/02S-1

COMPARING CLUSTERING ALGORITHMS FOR USE WITH GENOMIC AND PROTEOMIC DATA

THESIS

Rebecca Ann Olson
Civilian, USAF

AFIT/GAM/ENC/02S-1

AFIT/GAM/ENC/02S-1

COMPARING CLUSTERING ALGORITHMS FOR USE WITH GENOMIC AND PROTEOMIC DATA

THESIS

Presented to the Faculty
Department of Mathematics and Statistics
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Applied Mathematics

Rebecca Ann Olson, BS
Civilian, USAF

September, 2002

AFIT/GAM/ENC/02S-1

COMPARING CLUSTERING ALGORITHMS FOR USE WITH GENOMIC AND PROTEOMIC DATA

Rebecca Ann Olson, BS

Civilian, USAF

Approved:

/signed/	
Dennis W. Quinn (Chairman)	Date
/signed/	
Thomas F. Reid (Member)	Date
/signed/	
William P. Baker (Member)	Date

Acknowledgements

I would like to thank my advisor, Dr. Dennis W. Quinn, for his guidance and patience with me throughout the thesis effort. I would like to thank my co-worker, Thomas Hopkins, for helping with many different aspects of data processing and for putting up with me when I wanted to procrastinate.

I would also like to thank my family. My parents have been a tremendous support to me through many tough times over the past two years. I have also looked up to my little brother for quite some time, and he has inspired me to "keep up" with him.

Finally, I would like to thank my friend, Candace Thompson, and my roommate, Matthew Lange, for the times they provided me with stress relief.

Rebecca Ann Olson

Table of Contents

Page

List of Figures

List of Tables

AFIT/GAM/ENC/02S-1

Abstract

The Human Genome Project and related projects have resulted in the development of a number of new experimental and analytic tools for use in genomic and proteomic research. In the area of toxicogenomics, researchers are concerned with how genes react to exposure to certain chemicals.

The United States Air Force is interested in the effect of exposure to mission-essential chemicals. Although military personnel may come into contact with chemicals such as hydrazine, risk assessment is usually very limited. On the genomic level, risk assessment is a multi-step and multi-disciplinary process. The process begins with an experiment that exposes cells to the chemical. Data from the experiment are obtained using gene chips. The data can then be analyzed.

This research explores the methods of pre-processing and analyzing data. Several different data sets are used to compare the effectiveness of various clustering algorithms and their implementations. Genomic and proteomic data obtained from a hydrazine exposure experiment are then analyzed. A relationship is established between the genomic and proteomic data sets and is used in further analyses.

COMPARING CLUSTERING ALGORITHMS FOR USE WITH GENOMIC AND PROTEOMIC DATA

I. Introduction

1.1 Overview

The Human Genome Project and related projects have resulted in the development of a number of new experimental and analytic tools for use in genomic and proteomic research. Applications include, but are not limited to, disease diagnosis and prognosis, pharmacogenomics and toxicogenomics, classification of toxicants, and identification of biomarkers [4]. In the area of toxicogenomics, researchers are concerned with how genes react to exposure to certain chemicals. To assess the exposure reaction, experiments to observe genomic and proteomic levels during the exposure to a chemical are performed. Such experiments are costly in terms of time and money.

1.2 Problem

The United States Air Force is interested in the effect of exposure to mission-essential chemicals. Many chemicals readily used by the Air Force are not common to the civilian world and the civilian population does not usually come into contact with them. In such cases, risk assessment is usually very limited due to associated cost. In most cases, risk on a genomic scale has not been assessed at all. However, many military personnel may come into contact with chemicals such as hydrazine on a regular basis.

Assessing the risks of chemical exposure on the genomic level is a multi-step and multi-disciplinary process. The process begins with an experiment that exposes cells, namely rat hepatocytes, to the chemical. Data from the experiment are obtained using gene chips that are produced by Affymetrix. We can then analyze the data and attempt to make inferences about it.

1.3 Scope

This research explores the process of analyzing the data. The data can be processed in many different ways before it is actually analyzed. Also, a wide variety of clustering algorithms and implementations of those algorithms exist to assist in analyzing the data. In order to assess the effectiveness of the data preprocessing and the clustering algorithms, many different data sets were used. Several data sets were created in which the desired clustering results are known. Another data set, obtained with the software GeneCluster, has been used in previous research by Tamayo et al. [17]. Finally, genomic data obtained from the Air Force Research Laboratory (AFRL) were analyzed. The experiment generating the data exposed cells to hydrazine and will be explained further in section 3.5.

1.4 Approach

The approach of this research is fairly straightforward. The steps taken for analyzing the data are as follows:

1. Preprocess data to remove any inconsistencies.
2. Format data for appropriate clustering programs.
3. Normalize data if desired.
4. Run clustering programs.
5. Obtain and interpret results.

This approach was used to analyze several different data sets, and will be explained further in later chapters.

1.5 Summary of Thesis

This thesis is organized as follows:

Chapter II presents several clustering algorithms that are often implemented. It presents several implementations of those algorithms which are used in the course of this research. Genomics research involving the use of clustering algorithms is also discussed.

Chapter III presents several data sets which are used throughout the course of this research. It also explains the processing of the data that was done before the clustering algorithms could be applied to the data sets.

Chapter IV presents the results that were obtained from various clustering algorithms used to analyze several different data sets.

Chapter V summarizes the work that has been done, gives the conclusions that were reached, and gives recommendations for future work in this area.

II. Background

2.1 Overview

There are several ways to analyze the data in order to obtain meaningful results. These approaches are pattern recognition techniques and include, but are not limited to, cluster analysis, principal component analysis, and neural networks. For our purposes, I have concentrated primarily on cluster analysis. One of the difficulties that exist with clustering and interpreting the data is that we cannot visualize data that exists in higher dimensions and therefore clusters are not easily identified. Another common difficulty is that many data sets cannot be easily resolved into appropriate clusters based on close proximity of the points.

Cluster analysis has been explored for over 25 years. Through the years, the concepts in cluster analysis have changed, been refined, and grown. Today, algorithms can typically be classified into three different groups: hierarchical, partitioning, and competitive learning methods. Each type of algorithm has its strengths and weaknesses.

One of the most important characteristics of clustering methods is the definition of distance. In order to implement an algorithm, some sort of distance, similarity measure, or difference measure must be defined. This allows for the algorithm to specify what should be clustered together and when. The distances that are of concern are the distances between clusters, between observations, and between observations and clusters. Typically, the distance that is used is the Euclidean distance. The Euclidean distance between objects I and K which lie in N dimensional space is given by the equation [9:58]

$$D(I,K) = [\sum_{J=1}^{N}(A(I,J) - A(K,J))^2]^{\frac{1}{2}} \tag{2.1}$$

Other distance measures such as Minkowski, City-Block, and Mahalanobis may also be used, but Euclidean is the most well known [6:162]. Using a specific distance equation, the distance between observations is rather straightforward. However, there are still many different ways to calculate the distance between clusters. A cluster is simply a collection of observations, usually without any defined boundaries. To actually measure the distance between two different collections of observations creates a problem. The definition of the distance between clusters often determines the algorithm. Usually, the partitioning algorithms define the distance between clusters to be the distance between the centers of the clusters. Competitive learning methods take this a step farther and include a variable term in the calculation of the center of a cluster, in order to account for certain characteristics of the clusters. Hierarchical methods usually calculate the distance between clusters directly based on the observations within the cluster. These distances will be defined in greater detail as specific algorithms of these methods are described.

2.2 Hierarchical Algorithms

One of the oldest and most widely used algorithms, single linkage, belongs to the class of algorithms called hierarchical algorithms [9:191]. Hierarchical algorithms can be further classified as either agglomerative or divisive methods. Agglomerative methods generally follow the same form in that each object of the set begins as its own cluster. Then the method iteratively joins the two closest clusters into one cluster. An agglomerative method ends with one cluster containing all objects. The difference between various agglomerative methods is how the distance between clusters is defined. The most popular agglomerative method is the single linkage method [10:309]. This method defines the distance between clusters as the distance between the closest pair of objects, with a pair of objects containing one object from each cluster. This is also known as the distance between the nearest neighbors. Many resources have explained single linkage, however, occasionally some distinct differences in the descriptions arise. A version of the most widely accepted description of the single linkage algorithm can be found in the book *Cluster Analysis* by Brian Everitt [8:57–60]. Everitt's single linkage algorithm follows the general agglomerative algorithm using the nearest neighbor distance as previously described. The results are displayed in a dendrogram. The dendrogram is a visual representation of the hierarchy that occurred during the grouping of the observations. Clusters are obtained by making a cut in the dendrogram. The number of lines that are cut indicate the number of resulting clusters, and the members of each of those clusters are usually easy to identify from the dendrogram. Figure 2.1 shows a sample dendrogram from a test data set described in section 3.3. The horizontal dashed line shows where the dendrogram is cut resulting in four clusters. Following the the dendrogram down from the first cut on the left shows that one cluster includes observations 4, 6, 5, and 7. The other three clusters can be obtained in the same manner.

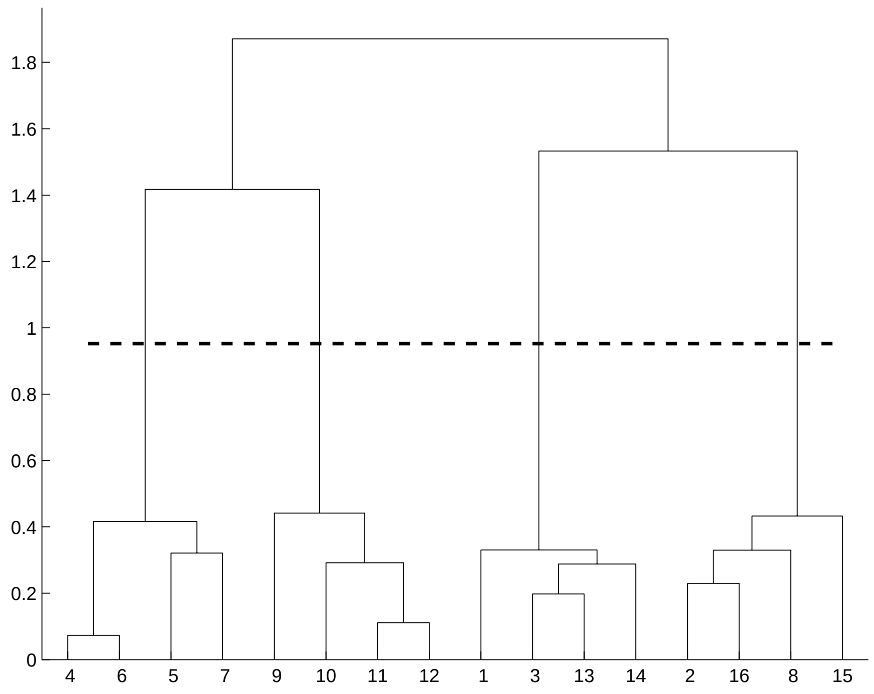


Figure 2.1. This graph shows a dendrogram of a test data set.

Everitt goes on to say that single linkage is closely related to minimum spanning trees in graph theory [8:60]. Another description of the single linkage algorithm, by John Hartigan in his book, *Clustering Algorithms*, is actually quite similar to minimum spanning trees [9:201], but different from the algorithm described by Everitt. Hartigan's single linkage algorithm and minimum spanning tree algorithm both choose an object arbitrarily and link it to its nearest neighbor. The next steps of the two algorithms iteratively join or link the closest neighbor to any of the objects already joined. In Hartigan's single linkage algorithm, the steps are repeated for each object. In the minimum spanning tree algorithm, the algorithm is finished when there are $M - 1$ links [9:201]. The results from Hartigan's single linkage algorithm are displayed in a visual graph similar to a dendrogram. However, the resulting tree is much more difficult to interpret [9:195]. In Hartigan's explanation of the minimum spanning tree algorithm he states that, "the order in which objects are added to the tree is the order in which the clusters are contiguous in the single-linkage algorithm," [9:201]. Therefore, both algorithms produce the same results. The difference occurs in how the two algorithms approach the problem. Hartigan's single linkage algorithm works through mathematical representation and displays the results visually, but the minimum spanning tree algorithm works through a more visual approach throughout the entire algorithm.

Hartigan's single linkage algorithm and minimum spanning tree algorithm differ from Everitt's single linkage algorithm in two key points. First, Everitt's algorithm begins by joining the two closest objects while Hartigan's algorithms choose an arbitrary object and find the closest object to it. Also, during the iterations, Hartigan's algorithms will only join the closest object to previously joined objects. Everitt's algorithm allows two objects to be joined in which neither objects have previously been joined. Implementation and use of these two single linkage algorithms can result in different clustering outcomes.

Complete linkage is very similar to single linkage. In complete linkage the distance between clusters is the distance between the farthest pair of objects, with a pair of objects containing one object from each cluster. This distance measure is often referred to as farthest neighbor.

Another widely used algorithm is average linkage, or group-average. In average linkage the distance between two clusters is defined to be the average distance between all pairs of objects in the two clusters. The formula for this calculation is given by [6:172]

$$\frac{1}{n_i n_j} \sum_{i=1}^{n_i} \sum_{j=1}^{n_j} d_{ij} \tag{2.2}$$

where d_{ij} is the distance between objects i and j, each belonging to a different cluster. The number of objects in the two clusters are represented by n_i and n_j. Often average linkage and single linkage will result in the same clusters. This is especially true in smaller data sets or data sets with clearly defined clusters.

In the centroid method, the centroid of the cluster is calculated in order to determine the distance between clusters. The centroid method is described well in Everitt [8:62]. This method represents a cluster by a mean vector, or centroid, and the distance between clusters is the distance between the mean vectors. The mean vector can be thought of as a point which represents the cluster but is not an actual observation. It is calculated by taking the mean of the corresponding coordinates of all the objects within the cluster. One disadvantage of this method is that when a small cluster is joined to a large cluster, the characteristic properties of the smaller cluster are lost [8:65]. Figure 2.2 gives an example of this. The circles indicate the centroids of the two separate clusters and are not actual data points. When the two clusters are joined, the new centroid remains within the large cluster, indicated by the triangle.

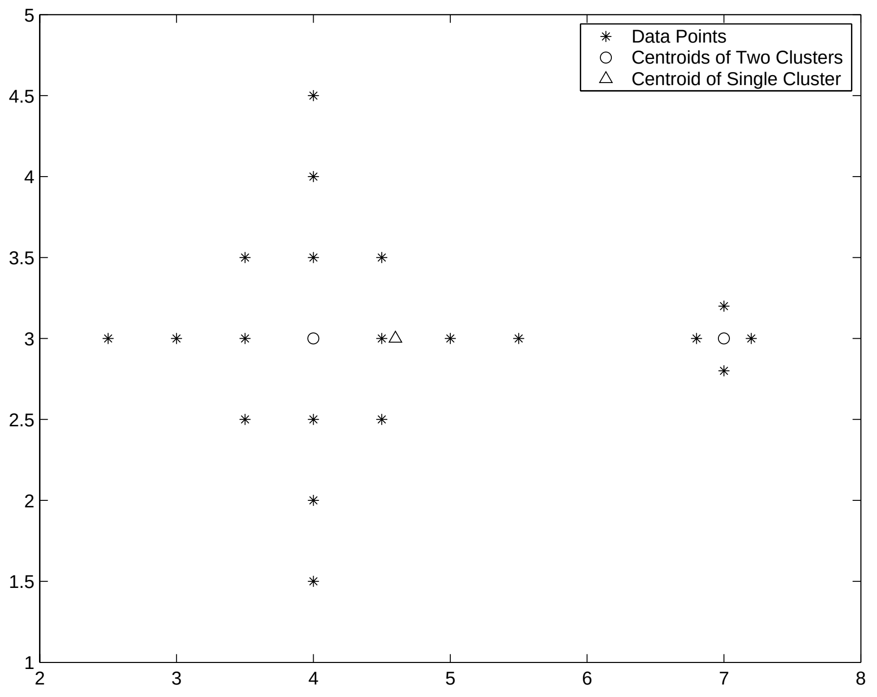


Figure 2.2. This graph shows an example of centroid clustering.

Another hierarchical method considered during this research is Ward's method. Ward's method calculates the error sum of squares given by the equation [19:237]

$$ESS = \sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2 \tag{2.3}$$

which reduces to the equation [8:66]

$$ESS = \sum_{i=1}^{n} (x_i - \bar{x})^{\frac{1}{2}} \tag{2.4}$$

where x_i is the i^{th} observation of the variable x, and $\bar{x}$ is the mean. Ward's method seeks to minimize this error without placing each observation in its own cluster. In addition to clustering by Ward's method, the error sum of squares has been used with other methods to help choose the appropriate number of clusters.

Divisive hierarchical methods are just the reverse of agglomerative hierarchical methods [13:59]. Instead of starting with the same number of clusters as objects and ending with one large cluster, divisive algorithms start with all the objects in one cluster and end with each object as its own cluster. Divisive methods generally partition the set into two roughly even clusters, and then iteratively partition each cluster into two clusters. The methods end when each cluster contains only a single object. There are $2^{N-1}-1$ possible partitions of the set into two clusters, where N is the number of objects in the set [6:178]. Divisive algorithms differ from one another by how they choose to partition the set. A disadvantage of divisive methods is that they are not as flexible as agglomerative methods [2:152].

Several useful hierarchical algorithms exist that have not been mentioned here. These algorithms, including divisive methods, were not considered during the course of this research due to disadvantages of the algorithms with respect to the type of data being used.

Some disadvantages to hierarchical methods are that they do not allow objects to change clusters once they have been assigned, and that they tend to chain objects together into existing clusters instead of creating new clusters [6:168, 186]. Sometimes the chaining affect is not considered a disadvantage, especially when the clusters should be chained together. An example of this is the case when the clusters have an elliptical shape. Since the algorithms link the closest objects together, the chaining affect describes how the algorithms work [8:68]. Advantages of the hierarchical methods are that they are easy to understand and implement.

2.3 Partitioning Methods

Partitioning methods differ from hierarchical methods in that the objects may be moved from one cluster to another as needed. Partitioning algorithms differ from each other by how the initial clusters are determined, how objects are assigned to clusters and how some or all of the objects assigned to clusters are reassigned to other clusters [6:186]. The k-means algorithm is a popular example of a partitioning method.

K-means is one of the most popular partitioning algorithms to use for clustering purposes [22:1]. The basic concept of the algorithm is to move objects from one cluster to another until the error component of the partition cannot be minimized further. It is similar to the centroid method in that the centroid of the cluster is calculated in order to determine the distance between objects and clusters. One disadvantage to the k-means algorithm is that the local optimum it converges to is sensitive to initialization [13, 22:97, 1]. The k-means algorithm selects an initial partition, generates a new partition by assigning an object to its closest centroid and then computes the new centroid of the cluster. The algorithm iterates until the error component is minimized [3:44]. The initial partition can be set up in many different ways. Some of the more popular ways to initialize include choosing the K objects that are furthest apart, choosing the first K objects in the data set, choosing cluster centers at intervals of one standard deviation on each variable, and choosing K and initial cluster centers based on prior knowledge if available [6:187]. The distance between the i^{th} object and the m^{th} cluster used in the algorithm is given by the equation [9:85]

$$D(i,m) = (\sum_{j=1}^{p}[x(i,j) - \bar{x}(m,j)]^2)^{\frac{1}{2}} \tag{2.5}$$

and the error component is given by the equation

$$E[P(n,K)] = \sum_{i=1}^{n} D[i,m(i)]^2 \tag{2.6}$$

where p is the dimension of the space, item i belongs to cluster $m(i)$, $\bar{x}(m,j)$ is the mean of the j^{th} variable in the m^{th} cluster, $x(i,j)$ is the value of the j^{th}variable for the i^{th} individual, $i = 1,2,\ldots,n$; $j = 1,2,\ldots,p$; and $P(n,K)$ is the partition that results in each of the n objects being allocated to one of the clusters $1,2,\ldots,K$. The error component is also referred to as the error sum of squares and has been used in this research to compare the clustering results of several different methods. The use of the error sum of squares will be discussed in later chapters.

2.4 Competitive Learning Methods

Competitive learning methods are claimed to be well suited for recognizing and classifying features in multidimensional data. If X is a collection of objects that we want to group into clusters with centroids w_j, then the goal of the competitive learning algorithm "is to move each of the centroids, w_j, to regions of the vector space that are 'dense' in vectors of X," [18:504]. The w_j's are referred to as representatives of the clusters. Several different methods exist but they can be described by one general algorithm. Sergios Theodoridis and Konstantinos Koutroumbas in their book, *Pattern Recognition*, describe the general idea of the algorithm:

> When a vector **x** is presented to the algorithm, all representatives *compete* with each other. The winner of this competition is the representative that lies closer (according to some distance measure) to **x**. Then, the winner is updated so as to move toward **x**, while the losers either remain unchanged or are updated toward **x** but at a much slower rate. [18:505]

The generalized algorithm for competitive learning methods is also presented by Theodoridis and Koutroumbas:

> Let t be the current iteration and t_{max} the maximum allowable number of iterations. Also, let m be the current number, m_{init} the initial number, and m_{max} the maximum allowable number of clusters (representatives). Then, a generalized competitive learning scheme (GCLS) may be stated as follows.
>
> *Generalized Competitive Learning Scheme(GCLS)*
>
> - $t = 0$
> - $m = m_{init}$
> - (A) Initialize any other necessary parameters (depending on specific scheme).
> - Repeat
>   - $t = t + 1$

```
    – Present a new randomly selected $\mathbf{x} \in X$ to the algorithm.
    – (B) Determine the winning representative $\mathbf{w}_j$.
    – (C) If (($\mathbf{x}$ is not "similar" to $\mathbf{w}_j$) OR (other condition)) AND
      ($m < m_{max}$) then
        * $m = m + 1$
        * $\mathbf{w}_m = \mathbf{x}$
    – Else
        * (D) *Parameter updating*
          $\mathbf{w}_j = \begin{cases} \mathbf{w}_j(t-1) + \eta h(\mathbf{x}, \mathbf{w}_j(t-1)), & \text{if } \mathbf{w}_j \text{ is the winner} \\ \mathbf{w}_j(t-1) + \eta' h(\mathbf{x}, \mathbf{w}_j(t-1)), & \text{otherwise} \end{cases}$
    – End
• (E) Until (convergence has occurred) OR ($t > t_{max}$)
• Identify the clusters represented by $\mathbf{w}_j$'s, by assigning each vector,
  $\mathbf{x} \in X$, to the cluster that corresponds to the centroid closest to $\mathbf{x}$.
```

The function $h(\mathbf{x}, \mathbf{w}_i)$ is an appropriately defined function. Also, η and η' are the *learning rates* controlling the updates of the winner and the losers, respectively. [18:505–506]

The most popular competitive learning method is the self-organizing map (SOM). The algorithm for the SOM is the same as the algorithm previously stated except for a change in part (D). In part (D), $h(\mathbf{x}, \mathbf{w}_j(t-1)) = \mathbf{x} - \mathbf{w}_j(t-1)$, $\eta' = 0$, and $\mathbf{w}_j$ is the winner if $\mathbf{w}_j \in Q_j(t)$ [18:511]. The variable learning rate, η, depends on t. The choice for $\eta(t)$, is crucial for convergence and typically has the constraints that $\eta(t)$ is a positive decreasing sequence that converges to zero, the summation of $\eta(t)$ as t goes from zero to infinity is equal to infinity, and the summation of $\eta^r(t)$ as t goes from zero to infinity is less than positive infinity for $r > 1$ [18:507]. $Q_j(t)$ is defined to be a neighborhood of representatives centered at $\mathbf{w}_j$ [18:511]. Further detail on the SOM and other competitive learning methods can be found in reference [18].

2.5 Implementation

The algorithms for the different cluster methods are relatively straightforward, however, two different implementations of the same algorithm can give different results. For this research several implementations of various methods were used. One of the most readily accessible software packages is Matlab. Using the statistical toolbox, Matlab has the capability to produce clustering results using single, average, complete, and centroid linkage methods. The statistical software package SAS also offers implementations of several hierarchical methods including single, average, complete and centroid linkage methods, as well as an implementation of the k-means algorithm. The program GeneCluster created by the Whitehead Institute at MIT is a useful implementation of the self-organizing map. Finally, Eisen's software Cluster offers single, average, and complete linkage, k-means, and self-organizing map. His TreeView software creates dendrograms to display the results from the hierarchical methods [7].

Both implementations in SAS and Matlab of the single linkage algorithm follow the algorithm previously described by Everitt. Although it is unknown how single linkage is implemented in Cluster, the results suggest it is implemented using the same algorithm as the other programs. After several trials using the single linkage algorithm with all three programs, it was determined that results using single linkage usually matched those given by average linkage or resulted in extensive chaining with no clearly defined clusters. For this reason, single linkage was not often used in this research. Instead, average and complete linkage were the primary focus in each of the hierarchical clustering programs.

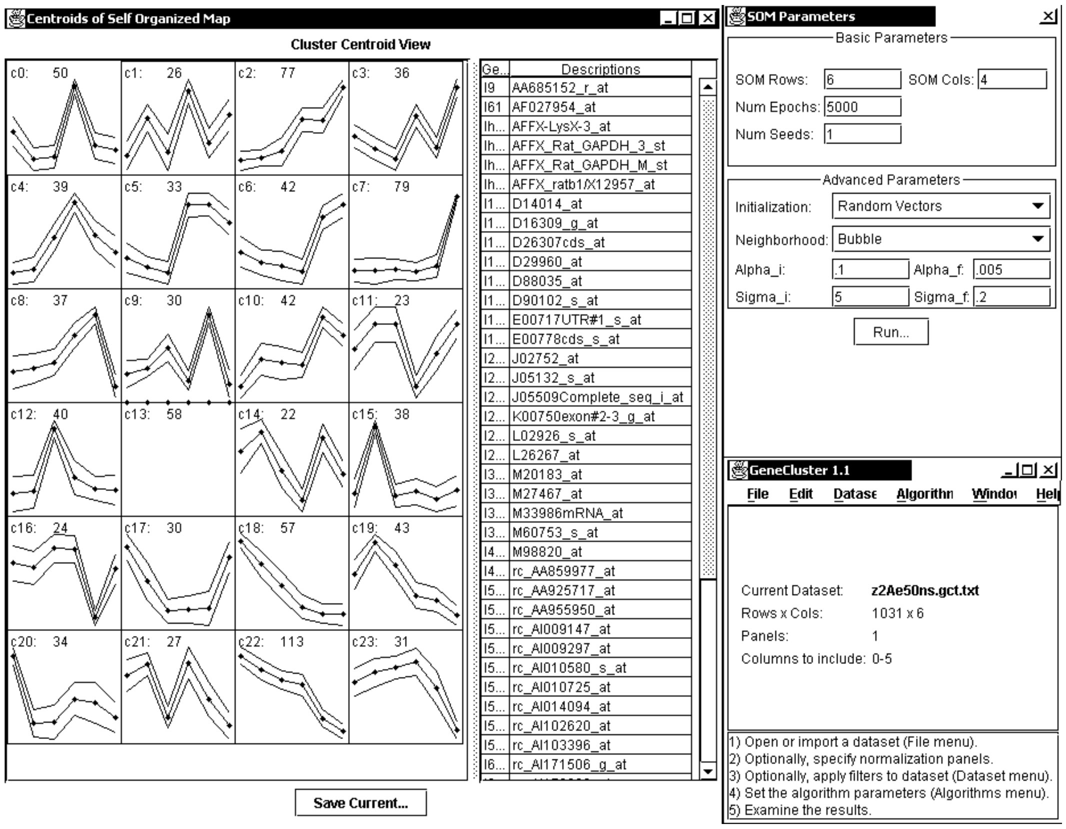
2.5.1 Matlab. Matlab was used to obtain clustering results using its average linkage and complete linkage hierarchical algorithm. Results can be obtained from Matlab in two ways. If the data set is small enough, the results can be interpreted from a dendrogram. For larger data sets, clusters can be specified by either a threshold or the number of clusters desired. The cluster information is separate from the data set, but can be saved, imported into Excel, and attached to the data set. For this reason, Matlab was more tedious to use than most of the other programs, but it also allowed more flexibility in interpreting the results.

2.5.2 Cluster. Eisen's Cluster program offers several different algorithms [7]. I used the self-organizing map (SOM), average linkage and complete linkage hierarchical, and k-means algorithms. The k-means algorithm asks for the number of clusters, but does not always create as many as the user specifies. I saved the results from trials that produced the number of clusters I specified. The k-means algorithm reorders the data so that subsequent rows of observations are in the same cluster and inserts an empty row between clusters. The results from the hierarchical algorithms have to be read into Eisen's TreeView program. TreeView creates a dendrogram that can be saved as an image and later manipulated in various imaging programs. I chose to open the image in Microsoft Paint. Within the Microsoft Paint program, I determined the cluster members by cutting the tree to give the number of clusters I wanted. The results from SOM are fairly easy to interpret. The SOM inserts a column that specifies the cluster number for each observation. In order to run the SOM, Cluster requires an input of the number of iterations. As the number of iterations is increased the results generally converge, but the time it takes for the algorithm to run is increased.

A problem with using Eisen's software is that the hierarchical results are only recognizable in the program TreeView. It is a tedious process to obtain results solely from the dendrogram. With a data set of over 1000 genes, it is nearly impossible to break the dendrogram into recognizable clusters. I was not able to establish a better method for determining the results of the hierarchical methods in Cluster. Therefore, Eisen's hierarchical methods in Cluster do not seem to be feasible for large data sets. Another problem encountered is that the implementations of the various algorithms in Eisen's Cluster program are unknown to us. Using the k-means algorithm, the results that were obtained are significantly different from the results obtained from other programs and from any known true clustering of the data. Without knowing the implementation of the algorithm, it is unclear if the inconsistent results are due to the implementation or the preparation of the data.

2.5.3 GeneCluster. The software package GeneCluster, which was developed at the Whitehead Institute at MIT, implements the SOM algorithm [20]. Similar to the SOM in Cluster, GeneCluster creates a file with an extra column containing the cluster number for each observation. It also creates a file listing the centroid coordinates for each cluster. Within the program, a graphical representation for all the clusters is created, but the image can only be saved by making an image of the computer screen. Figure 2.3 shows an example of this output.

One problem I encountered is that GeneCluster requires the input of the number of epochs. This is similar to inputting the iterations in Cluster. As the number of epochs or iterations are increased, the results become more consistent. On smaller data sets like our synthetic data, it is easy enough to figure out how many epochs and iterations are appropriate to obtain reasonable results. On larger data sets, the time requirement plays an important role in deciding how many epochs to input. As the number increases, the time it takes for the algorithm to finish also increases. Even inputting a very large number of epochs or iterations does not guarantee that subsequent clustering of the same data set will give the same results. It is clear that the size of the data set will directly affect the number of epochs chosen. The programs take longer per epoch or iteration on a larger set than a smaller one. Also, a larger data set will usually require more epochs or iterations to obtain consistent



Ge..	Descriptions
I9	AA685152_r_at
I61	AF027954_at
Ih...	AFFX-LysX-3_at
Ih...	AFFX_Rat_GAPDH_3_st
Ih...	AFFX_Rat_GAPDH_M_st
Ih...	AFFX_ratb1/X12957_at
I1...	D14014_at
I1...	D16309_g_at
I1...	D26307cds_at
I1...	D29960_at
I1...	D88035_at
I1...	D90102_s_at
I1...	E00717UTR#1_s_at
I1...	E00778cds_s_at
I2...	J02752_at
I2...	J05132_s_at
I2...	J05509Complete_seq_i_at
I2...	K00750exon#2-3_g_at
I2...	L02926_s_at
I2...	L26267_at
I3...	M20183_at
I3...	M27467_at
I3...	M33986mRNA_at
I3...	M60753_s_at
I4...	M98820_at
I4...	rc_AA859977_at
I5...	rc_AA925717_at
I5...	rc_AA955950_at
I5...	rc_AI009147_at
I5...	rc_AI009297_at
I5...	rc_AI010580_s_at
I5...	rc_AI010725_at
I5...	rc_AI014094_at
I5...	rc_AI102620_at
I5...	rc_AI103396_at
I6...	rc_AI171506_g_at

SOM Rows: 6 SOM Cols: 4
Num Epochs: 5000
Num Seeds: 1

Initialization: Random Vectors
Neighborhood: Bubble
Alpha_i: .1 Alpha_f: .005
Sigma_i: 5 Sigma_f: .2

Current Dataset: **z2Ae50ns.gct.txt**
Rows x Cols: 1031 x 6
Panels: 1
Columns to include: 0-5

1) Open or import a dataset (File menu).
2) Optionally, specify normalization panels.
3) Optionally, apply filters to dataset (Dataset menu).
4) Set the algorithm parameters (Algorithms menu).
5) Examine the results.

Figure 2.3. This shows an example of the output from GeneCluster.

results. This research did not approach the problem statistically, due to the number of factors involved.

Another concern using GeneCluster is inputting the "SOM Rows" and "SOM Columns." These inputs create the number of clusters in terms of the dimension of the output. For example, inputting six rows and four columns creates twenty-four clusters. The output shows twenty-four blocks, each block representing a cluster, configured with six rows and four columns. The configuration plays an important role in how the data are clustered. For a test case with twelve clusters, which will be described more fully in the later chapters, I tested the configurations 3×4, 4×3, 2×6, 6×2, 1×12, and 12×1. Immediately it was clear that a reverse order of the configuration does not make a difference in the outcome of the clustering; a configuration of 3×4 will give the same clusters as 4×3. The three different configurations did not give consistent results. On the test data set, configurations of 12×1 and 6×2 usually produced more accurate results than the 4×3 configuration. Over several runs of each of the configurations, 12×1 and 6×2 also gave more consistent results than 4×3. With a smaller test data set with six clusters, results using a configuration of 3×2 were more consistent and accurate than a configuration of 6×1. It was noted that the better configurations also caused the program to take a longer time to finish running.

2.5.4 SAS. SAS offers many algorithms but I chose to only look at various hierarchical methods and the k-means algorithm. Average and complete linkage algorithms were the primary focus of the SAS hierarchical methods. Other methods that SAS offers include Ward's method, centroid, flexible, median, and McQuitty. These methods were tested on three test data sets. The results from these methods were not better than the results from average and complete linkage and therefore were not pursued. SAS displays the results from the hierarchical algorithms in a dendrogram as well as recording them in a separate file. The file can be exported as an Excel worksheet. The letters "OB" and the order number represented each observation. The cluster number is listed in a separate column. The k-means algorithm also records the results in a separate file that can be exported as an Excel worksheet. The observations are left in the original order and the cluster number is listed in a separate column. Results from SAS for both k-means and hierarchical algorithms include columns containing other information, which we have not found useful thus far.

2.6 Genomics Research

While the use of cluster analysis has been used for years in various scientific areas, its use in the area of genomics has been limited. Recent developments of experimental and analytic tools for genomic research has allowed biologists to acquire large quantities of genomic data in a short time span. In order to obtain biological sense of the data, cluster analysis is beginning to be employed. Few articles have been published which discuss the use of clustering analysis to interpret patterns within genomic data. Of those that have been published, only a fraction of them deal with the human genome. In the article, *Genomic Analysis of Gene Expression in C. elegans* by Hill et al. cluster analysis was used to interpret patterns in genomic data from the nematode *C. elegans* [12]. Specifically, gene expression was analyzed from six stages in the life span of the nematode. Data were obtained using Affymetrix software and normalized to have a mean of zero and a variance of one. The data were then clustered using a self-organizing map. The article discusses specific examples in which a resulting cluster contains genes that are linked to a particular stage of development [12]. The SOM algorithm is also used to cluster genomic data by Tamayo et al. in the article *Interpreting Patterns of Gene Expression With Self-Organizing Maps: Methods and Application to Hematopoietic Differentiation* [17]. Gene expression from hematopoietic differentiation in four well studied models was analyzed. The data were normalized to have a mean of zero and a variance of one and clustered using the software package GeneCluster. It is demonstrated through the details of a specific cluster that the resulting "clusters correspond to patterns of clear biological relevance" [17:2910]. Several genes in the specified cluster were expected based on the expression profile of the specified cluster and understanding of hematopoietic differentiation. The authors also suggest that unexpected genes in a cluster may give insight into differentiation or suggest a previously unknown biological connection [17]. Alon et al. also uses cluster analysis with human genomic data in the article *Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays* [1]. The data obtained from Affymetrix came from several different cell types and from cancerous and non cancerous tissue. Normalization was done so that the data have a mean of zero and a magnitude of one. The algorithm used to cluster the data ordered the data into a binary tree similar to the hierarchical dendrogram. Clustering resulted in the cancerous and noncancerous tissue data separated into two different clusters and the data from different cell lines separated into different clusters [1]. All three articles show that clustering analysis can be a valuable tool for analysis of genomic data.

III. Pattern Recognition Applied to Several Data Sets

3.1 Overview

The data I am primarily interested in came from an experiment by AFRL that observed genomic and proteomic data from hepatocytes (liver cells) exposed to the chemical hydrazine. The proteomic results were provided by Dr. Frank Witzmann of Indiana University [21]. For each transcript there is an expression value for the beginning exposure, for two hours later when exposure ended, and for three, six, twelve, and twenty-four hours after the exposure ended. Three different exposure amounts were used: zero, fifty and seventy-five millimoles of hydrazine. The genomic information was captured on gene chips which were obtained from Affymetrix. Proteomic information was obtained using 2-D gel electrophoresis. Both genomic and proteomic data sets were in the form of a matrix of data.

My goal was to make sense of the data by using various clustering programs. A difficulty arises when comparing implementations of various clustering methods. Most clustering methods are designed in way that allows them to discern some patterns in the data better than other patterns. If the types of patterns in the data are known ahead of time, then picking the correct clustering method is straightforward. Most data is received without knowledge of the pattern, and therefore figuring out which method to use is a complex problem. "The shape of the clusters is not known until the clusters have been identified, and the clusters cannot be effectively identified unless the shapes are known," [5:2]. The shape of a cluster refers to the geometric form observed when the data is plotted. Two shapes that clusters may take are spheres or ellipsoids. In a spherical cluster, all data points on the edge of the cluster are roughly the same distance from the center of the cluster. The data points on the edge of an elliptical shaped cluster are farther from the center in two opposite directions and closer to the center in other directions. With a data set consisting of two parallel elliptical clusters, often programs will impose a spherical shape by forming a cluster at each end of the true clusters. Members of each of the true clusters reside in both of the imposed clusters [11]. Methods that are created to overcome this problem are often unable to resolve true spherical clusters. This is a typical problem in cluster analysis.

3.2 Preprocessing the Data

In order to use a clustering program on a data set, the data set must be formatted in such a way that the program will recognize and import the data. Each of the clustering programs that I wanted to use requires a different format. Microsoft Excel allows the manipulation of data sets into various formats. Therefore, each data set was imported into Excel, edited to a specific format and saved with the appropriate file extension.

Most of the formats into which we have placed the data sets generally have followed the same basic design. A data set usually has one or two columns of identifiers followed by several columns of data. The first row of the data set may or may not contain a header. The data sets were saved in a text (tab delimited) format with a specific extension. Data sets imported into Matlab must have the extension ".dat". The set must not have a header row and all of the identifiers must begin with a letter instead of a number. For use in GeneCluster, the data set must be saved with the extension ".gct". The first column in the header states the number of data rows and the second column states the number of data columns. The identifiers begin in row two in the same columns as the headers. The data columns begin in the third column. Eisen's Cluster program allows for more information to be stored with the data. However, for our purposes, we used a basic format. The first column in the header row indicates that the column contains identifiers. To indicate that the second column also contains identifiers, the second column of the header is required to contain "NAME" [7]. The rest of the columns in the header designate the data columns. We chose to save these data sets with the extension ".esn" to identify that the set was intended to be used with Eisen's Cluster program. The SAS software package is the exception to the general format that was used for the rest of the programs. The format used for SAS was much easier to set up than the others. We omitted any identifier columns, but carefully retained the order of the data until the identifiers could be reattached. The header designated the data columns, and the data set was saved as a regular Excel worksheet with the extension ".xls". After the data sets were formatted, they were ready for the various clustering programs.

3.3 Synthetic Data

To analyze the different clustering algorithms, several synthetic data sets were created for which we knew the correct clustering results. The first group of data sets consisted of seventy-two points in two-dimensional space. When plotted, the points made six clusters, each cluster containing twelve points and shaped like a '+' sign. Each arm of the '+' sign contained three points. The points were evenly spaced to create clusters that had a circular shape. The center of each cluster was located at the intersection of the arms of the '+' sign, and lacked a data point. The clusters have a radius of 0.48 and a distance of 0.04 between the edge points of two neighboring clusters. In this case, the edge point of a cluster is closer to the edge point of a neighboring cluster than it is to the nearest point within the same cluster. This data set is identified as set $A2$. Two more data sets, sets $B2$ and $C2$, were created in a similar manner. The clusters in data set $B2$ have a radius of 0.36 and a distance of 0.28 between neighboring clusters. Figure 3.1 shows data set $B2$. In data set $C2$, the clusters have a radius of 0.30 and a distance of 0.40 between neighboring clusters.

For each data set, the data points were identified and ordered in two ways. First, the data points were identified sequentially, such that the first twelve data points were in one cluster, the next twelve points were in a second cluster, etc. Then the order of the data points was randomized so that the data points fell into clusters

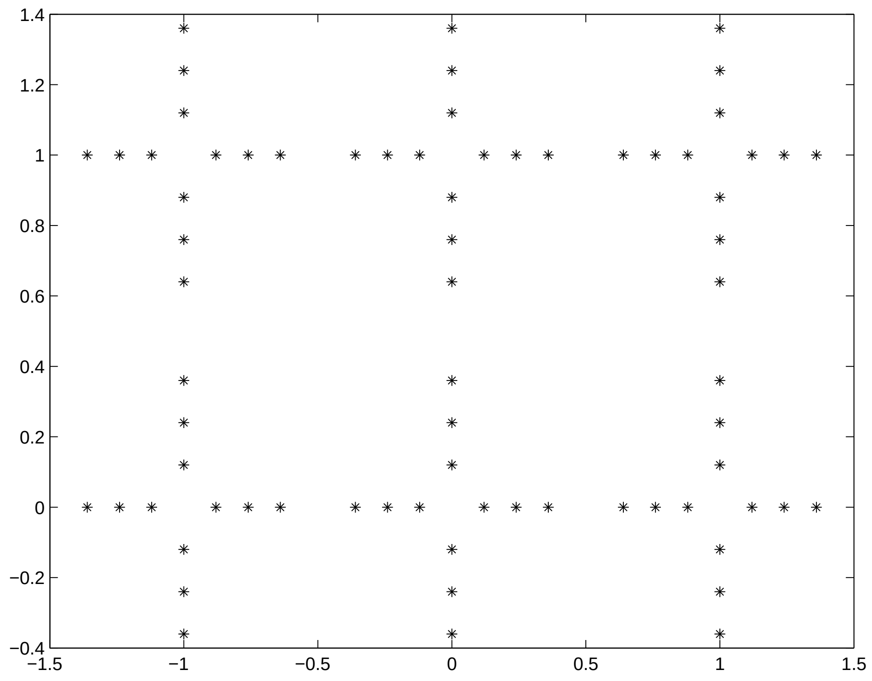


Figure 3.1. This graph shows the six clusters which make up the two-dimensional data set.

non-sequentially but with the same number of points per cluster. This ordering was intended to address the question of whether the ordering of a data set makes a difference in how the data are clustered.

After working with the circular cluster data, we decided to test how the same programs would cluster data that fell into elliptical shaped clusters in two-dimensions. Elliptical shaped clusters have the characteristic that a point at one of the elongated ends of a cluster is closer to some of the points in the cluster above or below it, the cluster beside it, and the cluster diagonal to it than it is to a point in the same cluster at the opposite elongated end. Programs that give good results with both circular and elliptical data sets can be considered more robust and would be more helpful in clustering genomic and proteomic data. Figure 3.2 shows a plot of the elliptical data set that was created. Each cluster had a horizontal radius of 0.36 and a vertical radius of 0.18. This data set was identified as set B_e2.

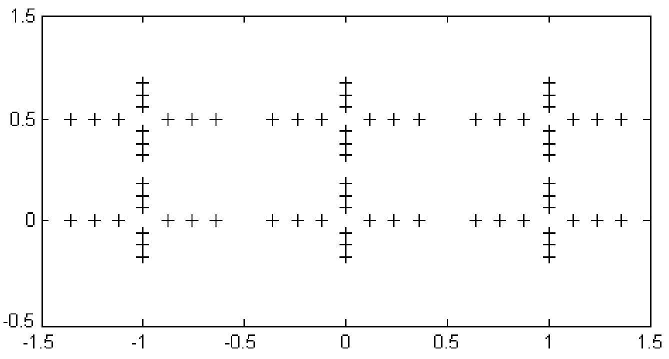


Figure 3.2. This graph shows the six clusters which make up the elliptical two-dimensional data set.

To get an idea of how these programs would work on higher dimensional data, we also produced a series of five data sets in three dimensions consisting of 216 points each. The 216 points are grouped into twelve spherical clusters, each cluster containing eighteen points. Similar to the two dimensional data, the three dimensional clusters were formed as a '+' sign, with an extra set of arms extending into the third dimension. The first set in the three-dimensional series was identified as set $A3$. It contained well defined clusters with the distance between the edges of neighboring clusters being longer than the distance between neighboring points of the same cluster. The clusters have a radius of 0.24. Figure 3.3 shows data set $A3$. The other four data sets in the series were identified as data sets $B3$, $C3$, $D3$, and $E3$. Each set successively shifted the clusters closer together. The radii of the clusters in sets $B3$ through $E3$ are 0.30, 0.36, 0.42, and 0.48. A distance of 0.04 separates the clusters in data set $E3$.

The synthetic data sets are simple examples of lower dimensional data. However, the genomic and proteomic data that will be discussed in later sections have a higher dimensionality. Specifically, several observations were made over a period of time. For this reason, another synthetic data set was developed to mimic a time series data set. A total of sixteen observations were created, each with four time steps. The sixteen observations fell into four clusters, each cluster containing four observations. Figure 3.4 shows the plot of the data. Each line represents a gene. The

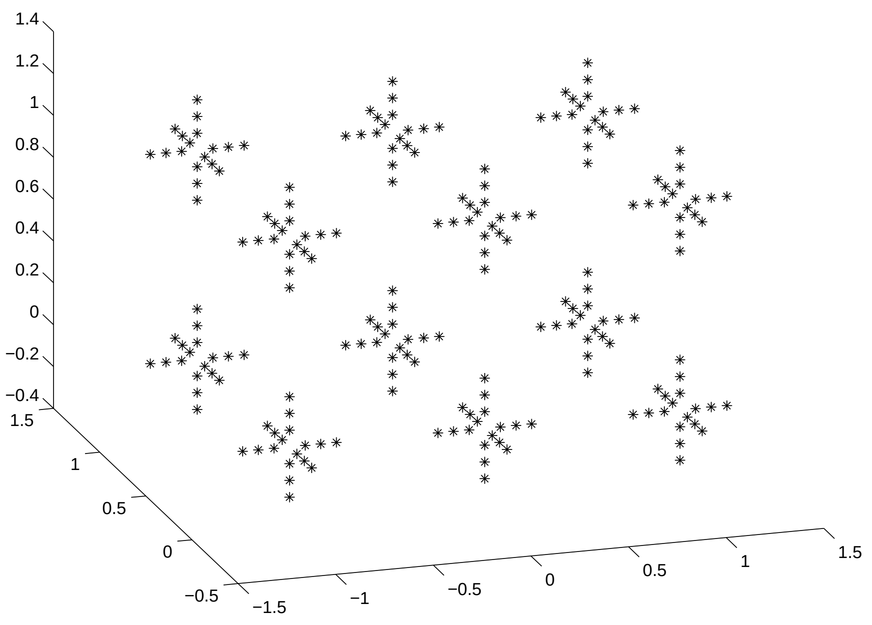


Figure 3.3. This graph shows the twelve clusters which make up the three-dimensional data set.

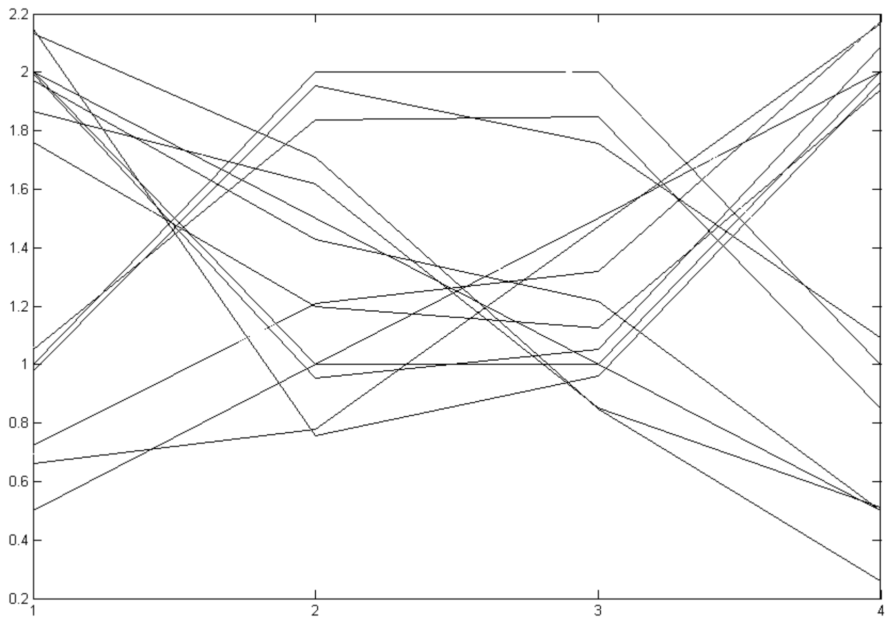


Figure 3.4. This graph shows the sixteen observations created in four-dimensional space.

vertical axis gives the value of the gene plotted at each time step on the horizontal axis. Figure 3.5 shows a separate plot for each of the four clusters, plotted in the same manner. Visually, it may be hard to distinguish the four clusters in Figure 3.4. However, it is clear from Figure 3.5 that four distinct clusters exist.

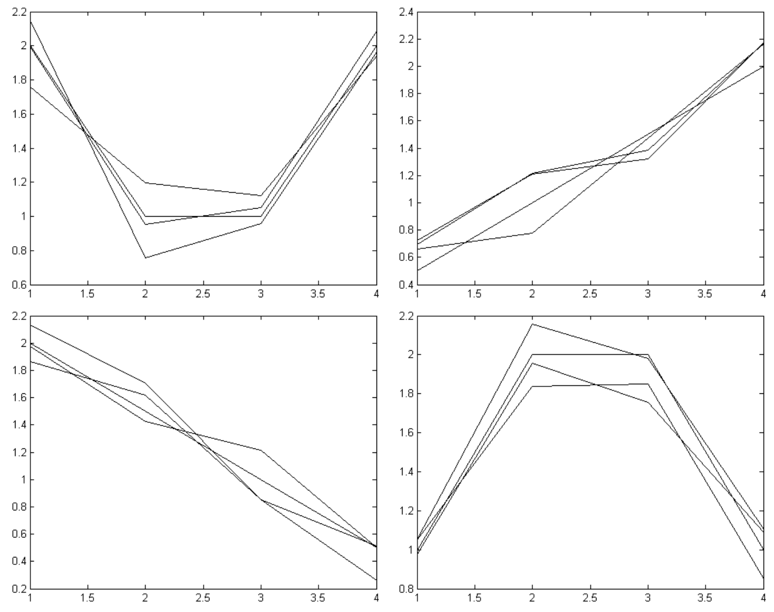


Figure 3.5. This graph shows the four distinct clusters of the synthetic four-dimensional data.

A script was written in Matlab to create each of the specified data sets. The script ordered the data in both of the ways previously described and plotted the points. This was done completely separate from using clustering programs within Matlab. The resulting data sets were saved and imported in Excel. Each was formatted and saved as specified by the various clustering programs.

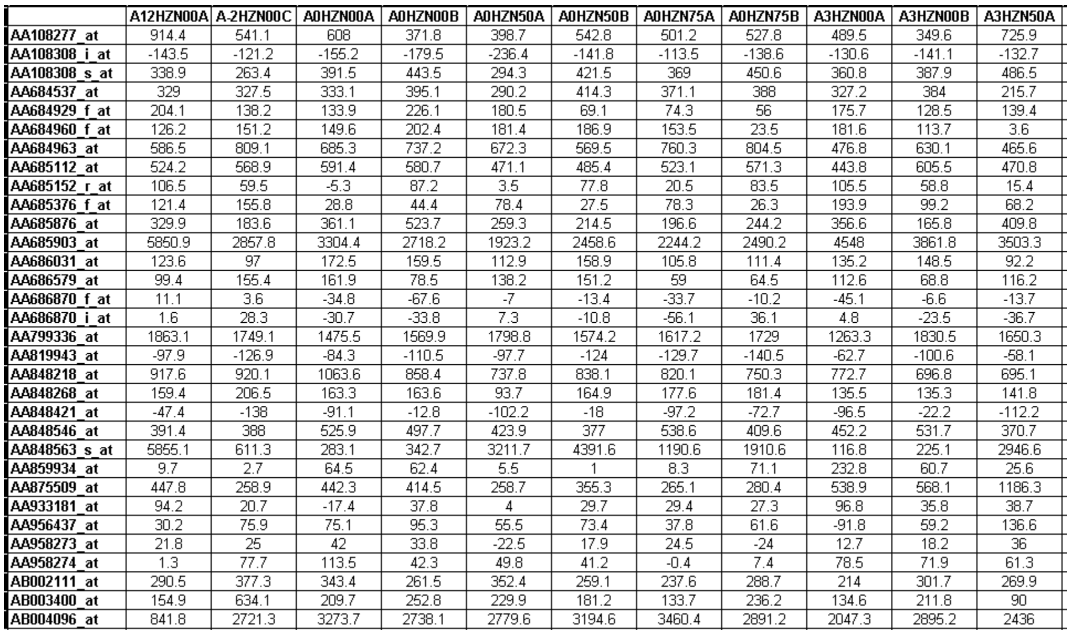
3.4 GeneCluster Data

Another data set we tested was provided with the software GeneCluster and is discussed in the article, *Interpreting Patterns of Gene Expression with Self-organizing Maps: Methods and Application to Hematopoietic Differentiation*, [17]. The data set consists of over 7,000 genes, each with expression levels at four time steps. Within

GeneCluster, filters are available to manipulate the data and reduce the data set. The authors used three different filters within GeneCluster before clustering. The first filter changed any data point that had an expression level less than 20 to be equal to 20. Any genes that did not change by a factor of three over the four time steps were eliminated by the second filter. This reduced the data set to less than 600 genes. Finally, the third filter normalized the data. By following these same steps as outlined in their paper, I was able to reproduce their results using GeneCluster. This is discussed further in section 4.2. The reduced data set was saved and formatted to fit the constraints of other software programs. I was able to reproduce the GeneCluster results using several other programs. This is also discussed in section 4.2. To check against bias that may have been introduced by GeneCluster, we compared the first and last fifty genes in the reduced data set to the original data set. It was clear that GeneCluster preserved the original order of the data during the reduction.

3.5 Genomic Data

The experiment conducted by AFRL generating the genomic and proteomic data was designed to take data from three different exposure levels of hydrazine at six different time steps. The time steps measured the passing of time during the experiment, with respect to the point when the genes were exposed to hydrazine. The time steps occur at the beginning of exposure, two hours later at the end of exposure, and three, six, twelve and twenty-four hours after exposure. Exposure was measured with three levels: zero, fifty, and seventy-five millimoles. Data was taken twice for each combination of an exposure level at a specific time step. The exceptions to this were that the time corresponding to the beginning of exposure did not have exposure levels of fifty or seventy-five. Also, for the zero exposure at six hours data were only taken once. This gave us a total of thirty-one columns of data. Each time the data was taken twice (two replications), the data from the two trials were averaged together. Figure 3.6 is a small portion of the original data before it was preprocessed. The first column contains the individual gene identifiers. The first row indicates which observation data is in the column. We wanted to look at the data with respect to the individual time step and the individual exposure levels. Three subsets of data were created, each corresponding to an exposure level. Each data subset contained six columns of data corresponding to the six time steps. The expression level for the beginning of exposure is the same for all three data sets. This is justified since the data should be the same for all exposure levels if no exposure has taken place. Five sets of data corresponding to different time steps were also created. Each subset had three data columns corresponding to the three exposure levels. A file was not created for the before exposure time step, since it would have only contained one column of data without exposure levels fifty and seventy-five. Each data set was saved as a separate file and formatted for use in the various clustering programs.



	A12HZN00A	A-2HZN00C	A0HZN00A	A0HZN00B	A0HZN50A	A0HZN50B	A0HZN75A	A0HZN75B	A3HZN00A	A3HZN00B	A3HZN50A
AA108277_at	914.4	541.1	608	371.8	398.7	542.8	501.2	527.8	489.5	349.6	725.9
AA108308_i_at	-143.5	-121.2	-155.2	-179.5	-236.4	-141.8	-113.5	-138.6	-130.6	-141.1	-132.7
AA108308_s_at	338.9	263.4	391.5	443.5	294.3	421.5	369	450.6	360.8	387.9	486.5
AA684537_at	329	327.5	333.1	395.1	290.2	414.3	371.1	388	327.2	384	215.7
AA684929_f_at	204.1	138.2	133.9	226.1	180.5	69.1	74.3	56	175.7	128.5	139.4
AA684960_f_at	126.2	151.2	149.6	202.4	181.4	186.9	153.5	23.5	181.6	113.7	3.6
AA684963_at	586.5	809.1	685.3	737.2	672.3	569.5	760.3	804.5	476.8	630.1	465.6
AA685112_at	524.2	568.9	591.4	580.7	471.1	485.4	523.1	571.3	443.8	605.5	470.8
AA685152_r_at	106.5	59.5	-5.3	87.2	3.5	77.8	20.5	83.5	105.5	58.8	15.4
AA685376_f_at	121.4	155.8	28.8	44.4	78.4	27.5	78.3	26.3	193.9	99.2	68.2
AA685876_at	329.9	183.6	361.1	523.7	259.3	214.5	196.6	244.2	356.6	165.8	409.8
AA685903_at	5850.9	2857.8	3304.4	2718.2	1923.2	2458.6	2244.2	2490.2	4548	3861.8	3503.3
AA686031_at	123.6	97	172.5	159.5	112.9	158.9	105.8	111.4	135.2	148.5	92.2
AA686579_at	99.4	155.4	161.9	78.5	138.2	151.2	59	64.5	112.6	68.8	116.2
AA686870_f_at	11.1	3.6	-34.8	-67.6	-7	-13.4	-33.7	-10.2	-45.1	-6.6	-13.7
AA686870_i_at	1.6	28.3	-30.7	-33.8	7.3	-10.8	-56.1	36.1	4.8	-23.5	-36.7
AA799336_at	1863.1	1749.1	1475.5	1569.9	1798.8	1574.2	1617.2	1729	1263.3	1830.5	1650.3
AA819943_at	-97.9	-126.9	-84.3	-110.5	-97.7	-124	-129.7	-140.5	-62.7	-100.6	-58.1
AA848218_at	917.6	920.1	1063.6	858.4	737.8	838.1	820.1	750.3	772.7	696.8	695.1
AA848268_at	159.4	206.5	163.3	163.6	93.7	164.9	177.6	181.4	135.5	135.3	141.8
AA848421_at	-47.4	-138	-91.1	-12.8	-102.2	-18	-97.2	-72.7	-96.5	-22.2	-112.2
AA848546_at	391.4	388	525.9	497.7	423.9	377	538.6	409.6	452.2	531.7	370.7
AA848563_s_at	5855.1	611.3	283.1	342.7	3211.7	4391.6	1190.6	1910.6	116.8	225.1	2946.6
AA859934_at	9.7	2.7	64.5	62.4	5.5	1	8.3	71.1	232.8	60.7	25.6
AA875509_at	447.8	258.9	442.3	414.5	258.7	355.3	265.1	280.4	538.9	568.1	1186.3
AA933181_at	94.2	20.7	-17.4	37.8	4	29.7	29.4	27.3	96.8	35.8	38.7
AA956437_at	30.2	75.9	75.1	95.3	55.5	73.4	37.8	61.6	-91.8	59.2	136.6
AA958273_at	21.8	25	42	33.8	-22.5	17.9	24.5	-24	12.7	18.2	36
AA958274_at	1.3	77.7	113.5	42.3	49.8	41.2	-0.4	7.4	78.5	71.9	61.3
AB002111_at	290.5	377.3	343.4	261.5	352.4	259.1	237.6	288.7	214	301.7	269.9
AB003400_at	154.9	634.1	209.7	252.8	229.9	181.2	133.7	236.2	134.6	211.8	90
AB004096_at	841.8	2721.3	3273.7	2738.1	2779.6	3194.6	3460.4	2891.2	2047.3	2895.2	2436

Figure 3.6. A small sample of the genomic data before preprocessing.

Several of the files were analyzed using GeneCluster. Twenty-four clusters were chosen and input as 6×4. The epochs and other settings in GeneCluster were left as the default. The resulting graphical representation of the clusters showed wide error bars around the centroid. Also, many clusters contained only a few observations while a few clusters contained several hundred observations. The graphical representation can be seen in Figure 3.7. Each box contains a different cluster and the centroids are represented by dotted lines. The number in the top center of each box indicates how many genes are in that cluster. The last cluster, in the lower right corner of the figure, contains over thirty percent of the total number of genes. The wide error bars around the centroid of this cluster indicate that the genes do not behave similarly despite being clustered together. These results were not very promising and will be discussed in more detail in chapter IV. After looking closer at the data, it was noticed that the data covered a wide range of values. There were negative values and values ranging in the tens of thousands. It was then decided that the data should be normalized to reduce the variability within the data. It was determined that negative values did not make sense, and that they should be changed to the value zero. This was done before normalizing. I chose to do a normalization for each transcript using the following equation:

$$S(x_i) = \frac{(x_i - \bar{x})}{(\sum_{j=1}^{p}(x_j - \bar{x})^2)^{\frac{1}{2}}} \tag{3.1}$$

where x_i is the i^{th} observation for the transcript, $\bar{x} = \frac{\sum_{j=1}^{n} x_j}{n}$, and p is the dimension of the space, in our case $p = 6$. The data were normalized within the files that were created. The new normalized data were saved as a separate file and formatted for use in the various clustering programs. Note that if $\mathbf{y} = c \cdot \mathbf{x}$ then $S(y_i) = S(x_i)$, that is, normalizing by equation 3.1 maps observations from distinct transcripts that are scalar multiples of each other onto the same points.

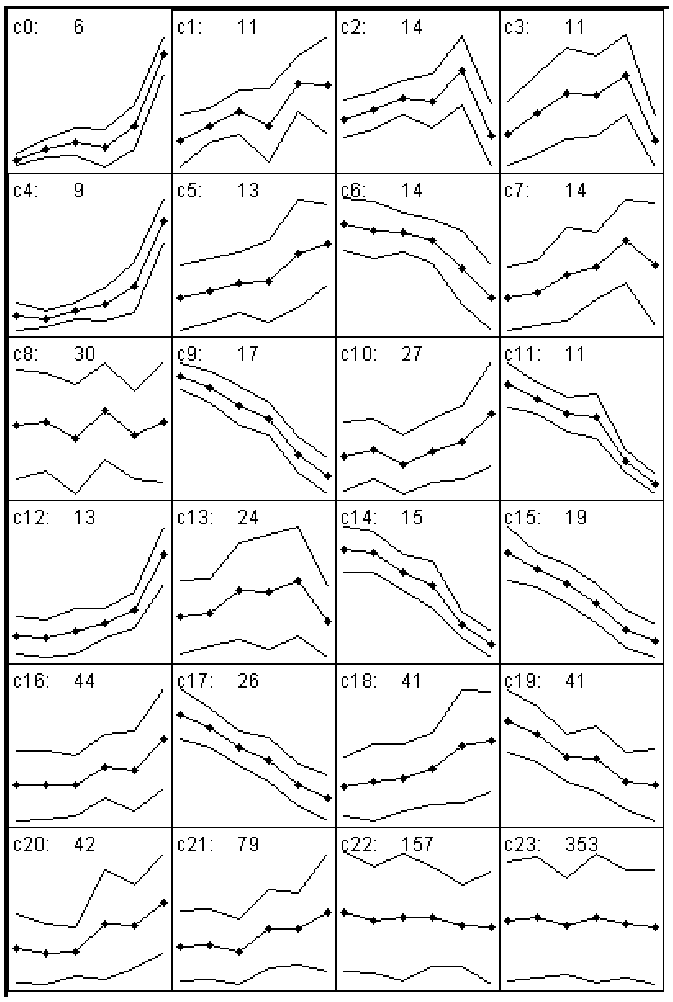


Figure 3.7. This visual output from GeneCluster was obtained using a genomic data file.

The normalized data files were run through GeneCluster, again using twenty-four clusters and the default settings. The resulting graphical representation of the clusters looked much better. The error bars were closer to the centroid and the data points were distributed throughout the clusters fairly evenly. Other clustering programs were then put to the test. Matlab was the second clustering program to be used. In Matlab, single linkage was used first. The resulting dendrogram was impossible to interpret with respect to individual genes. From the dendrogram, we were able to determine that Matlab's single linkage algorithm did not resolve distinct clusters. Instead, single linkage chained everything together.

Due to the chaining tendency of Matlab's single linkage algorithm, we decided to break the data into quartiles. A specific subset of the data was sorted by the row mean, then the top 258 normalized genes were copied and saved to a separate file. This was done with the next 258 genes and the bottom two sets of 258 genes. Each of these four files was formatted specifically for use in Matlab. Within Matlab, the single linkage algorithm was used on each of the four files. Even normalized data broken down into much smaller subsets still continued to result in a chaining effect without any distinct clusters. At this point, it was decided that single linkage was not appropriate for our data, and average and complete linkage algorithms were explored.

Several other subsets of the genomic data set were created. These files consisted of taking the data from the fifty millimole exposure file and subtracting from it the data from the zero exposure file. The data in this file were saved and formatted. Then a separate file was created with normalized data. This process was repeated with the zero exposure data subtracted from the seventy-five millimole exposure data, and with the fifty millimole exposure data subtracted from the seventy-five millimole exposure data.

3.6 Proteomic Data

The samples from the hydrazine exposure were also used to obtain proteomic data using 2-D gel electrophoresis. The resulting data sets were similar to the genomic data sets. However the proteomic data did not include multiple trials that needed to be averaged together or contain negative numbers. The data set contained only 141 proteins. It was then learned that multiple observations occurred for the same protein. This has an additive affect. The multiple observations were added together in the data set. This left 110 different protein observations. Just as with the genomic data, each protein observation included six time steps and three exposure levels. The data were separated into three different subsets, each corresponding to a different exposure level. Once separated into subsets, two different files were created and formatted for each exposure. One file contained the raw data as it was received, the other contained normalized data, as was previously described. The six files were formatted to be used in the various clustering programs.

3.7 Gene/Protein Relation

Each of the genomic and proteomic data sets were accompanied with identifiers and descriptions for the different genes and proteins. Since both sets came from the same experiment, there should be a relationship between the proteins and genes that were present. In order to find related genes and proteins within our data sets, we turned to the *SWISS-PROT Protein Knowledgebase TrEMBL Computer-annotated Supplement to SWISS-PROT* Web site [16]. At the Web site I specified a protein EMBL identifier which accompanied our protein data and was transferred to a web page that listed associated GenBank numbers. After obtaining GenBank associations for the proteins, we compared them to the genes in our genomic data set. We were able to find twenty-one genes that corresponded to sixteen proteins. Since more than one GenBank number can be associated with a single protein, we obtained more genes than proteins in the association between the two data sets. The newly identified genes and proteins were copied and saved into separate files. These files were normalized. I was interested in seeing if the related genes and proteins, in their separate files, would cluster similarly. Both sets of files were clustered using Matlab average linkage and using GeneCluster.

IV. Results

4.1 Synthetic Data

Given simple data sets in which the true clusters are known, it is easy to establish the accuracy of various clustering programs using these data. When accurate results were not obtained, most programs often produced clusters that were similar to the true clusters. One estimate of the accuracy of a program was to simply count how many points were put into the incorrect cluster. Table 4.1 shows the results from the two-dimensional data sets, including the elliptical set. The true clusters are located closest together in data set $A2$ and farthest in data set $C2$. Data set B_e2 is the elliptical data set that was created from data set $B2$. During the clustering pro-

Table 4.1. Count of incorrectly clustered points using 2-D synthetic data sets.

Program	Method	$A2$	$B2$	$C2$	B_e2
Matlab	Average	17	0	0	0
Matlab	Complete	7	0	0	36
Cluster	Average	13	9	9	37
Cluster	Complete	30(7)	9	9	12(7)
Cluster	SOM (6×1)	25	23	15	20
GeneCluster	SOM (2×3)	5	0	0	0
GeneCluster	SOM (6×1)	6	0	0	37
SAS	Average	17	0	0	0
SAS	Complete	7	0	0	36
SAS	K-means	10	0	0	16

cess, the algorithms are suppose to cluster together the two objects with the smallest distance between them. Due to the uniformity of the placement of points within a cluster, often the algorithm would find three or more objects that were equally as close to one another. In this case, all objects would be clustered together in the same step. In some cases, multiple clusterings within one step caused the correct number of clusters to be unobtainable. When this occurred, having an extra cluster was often more appropriate than having one cluster less than the desired amount. A small cluster that allowed for a close representation of the true clusters was chosen to be the extra cluster, and all the points of that cluster were counted as incorrectly clustered. This problem is indicated in the following tables by a number in parentheses that specifies the number of clusters that was found. Cluster's k-means algorithm was also used but the results were not comparable to the true clusters by use of a count of incorrectly clustered points. Therefore, the results have been left out of the table.

Similar results were obtained for the three-dimensional data sets. Table 4.2 gives these results. The true clusters are located closest together in data set $E3$ and farthest apart in data set $A3$. Cluster gave poor results with data sets $A3$, $B3$, and $E3$ compared to the other programs. Therefore, since Cluster is not appropriate for these kinds of data, data sets $C3$ and $D3$ using Cluster were not included.

Table 4.2. Count of incorrectly clustered points using 3-D synthetic data sets.

Program	Method	$A3$	$B3$	$C3$	$D3$	$E3$
Matlab	Average	0	0	0	40	20
Matlab	Complete	0	0	0	54	20
Cluster	Average	15(13)	15(13)			33(13)
Cluster	Complete	15(13)	15(13)			49
GeneCluster	SOM (3×4)	18(11)	18(11)	19(11)	20(11)	8
GeneCluster	SOM (6×2)	0	0	0	0	7
GeneCluster	SOM (12×1)	0	0	0	0	4
SAS	Average	0	0	0	26	20
SAS	Complete	0	0	0	31	20
SAS	K-means	0	0	0	32	15

From Tables 4.1 and 4.2, GeneCluster consistently gives the best results over all the data sets. However, in order to get the best results, the cluster configuration must be chosen correctly. Both tables show that Matlab and SAS give similar results when using the same linkage method. When using the same linkage methods in Cluster, the results are significantly different and are not close to the true clusters.

Counting the incorrectly clustered points is a simple way to gage the accuracy of a program. In the cases where the number of created clusters did not conform to the number of true clusters, this method does not give a clear picture of how the clustering method actually performed. Results from a large data set with higher dimensions would be very difficult to analyze in this manner. Another method of comparing results is to calculate the error sum of squares, as given in equation 2.6 in section 2.3. This method allows comparison of results using larger, multidimensional data sets from any clustering method. A small value for the error sum of squares indicates good clustering results. The error sum of squares was used in comparing the clustering results from both the two-dimensional and three-dimensional synthetic data sets. Table 4.3 gives the results for the two-dimensional data sets while Table 4.4 gives the results for the three-dimensional data sets. In most cases as the number of

Table 4.3. Error sum of squares for the 2-D synthetic data sets.

Program	Method	$A2$	$B2$	$C2$	B_e2
Matlab	Average	12.246	4.838	3.360	3.024
Matlab	Complete	8.504	4.38	3.360	6.142
Cluster	Average	10.670	8.306	7.488	8.961
Cluster	Complete	12.635	8.306	7.488	5.473
Cluster	SOM (6×1)	11.954	10.055	8.832	7.529
GeneCluster	SOM (2×3)	8.511	4.838	3.360	3.024
GeneCluster	SOM (6×1)	8.506	4.838	3.360	6.540
SAS	Average	12.246	4.838	3.360	3.024
SAS	Complete	8.504	4.838	3.360	6.142
SAS	K-means	9.140	4.838	3.360	4.357

incorrectly clustered points increases, the error sum of squares also increases. This can be seen by comparing the values in Table 4.1 with those in Table 4.3. The error sum of squares provides a better comparison in cases when the clustering method did not create the correct number of clusters. A good example of this occurs with data set $D3$ of the three-dimensional data. Using GeneCluster with a 3×4 configuration, eleven clusters were obtained with twenty incorrectly clustered points. The error

Table 4.4. Error sum of squares for the 3-D synthetic data sets.

Program	Method	$A3$	$B3$	$C3$	$D3$	$E3$
Matlab	Average	6.451	10.080	14.515	28.942	26.154
Matlab	Complete	6.451	10.080	14.515	41.683	26.154
Cluster	Average	13.248	15.943			33.730
Cluster	Complete	13.248	15.943			40.171
GeneCluster	SOM (3×4)	15.451	19.080	23.495	28.282	25.794
GeneCluster	SOM (6×2)	6.451	10.080	14.515	19.757	25.791
GeneCluster	SOM (12×1)	6.451	10.080	14.515	19.757	25.802
SAS	Average	6.451	10.080	14.515	26.118	26.154
SAS	Complete	6.451	10.080	14.515	27.553	26.154
SAS	K-means	6.451	10.080	14.515	29.248	26.761

sum of squares for this result is 28.282. A similar error sum of squares value was obtained using average linkage in Matlab which returned forty incorrectly clustered points. The error sum of squares values indicate that mathematically the results are similar.

Before clustering, most of the data were normalized. However, normalizing the two- and three-dimensional data sets removes important information. When the two-dimensional data were normalized, all of the data points above the line $y = x$ were mapped to the point $(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. Data points below the line $y = x$ were mapped to the point $(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$. Normalizing the three-dimensional data maps the data onto the unit circle, shown in Figure 4.1, and the true clusters are not distinguishable. Normalizing causes the sum of the observations for a transcript to equal zero, that is

$$\sum_{i=1}^{n} S(x_i) = 0 \tag{4.1}$$

where $S(x_i)$ is defined in equation 3.1 and n is the dimension of the space. If we let $y_i = S(x_i)$ we obtain $\sum_{i=1}^{n} y_i = 0$ and we can solve for y_n to obtain

$$y_n = -y_1 - y_2 - \cdots - y_{n-1} \tag{4.2}$$

From equation 4.2 it is clear that the observations are not linearly independent. This implies that normalizing reduces the dimensionality of the data to $n-1$. The data lie on an $n-1$ dimensional hyperplane in n dimensional space. Moreover $\sum_{i=1}^{n} y_i^2 = 1$ so the normalized data lie on a unit hypersphere on the $n-1$ dimensional hyperplane. The data points in the plot of Figure 4.1 all lie on a two-dimensional plane. Similarly, the normalized two-dimensional data can be plotted as two points on a one-dimensional line.

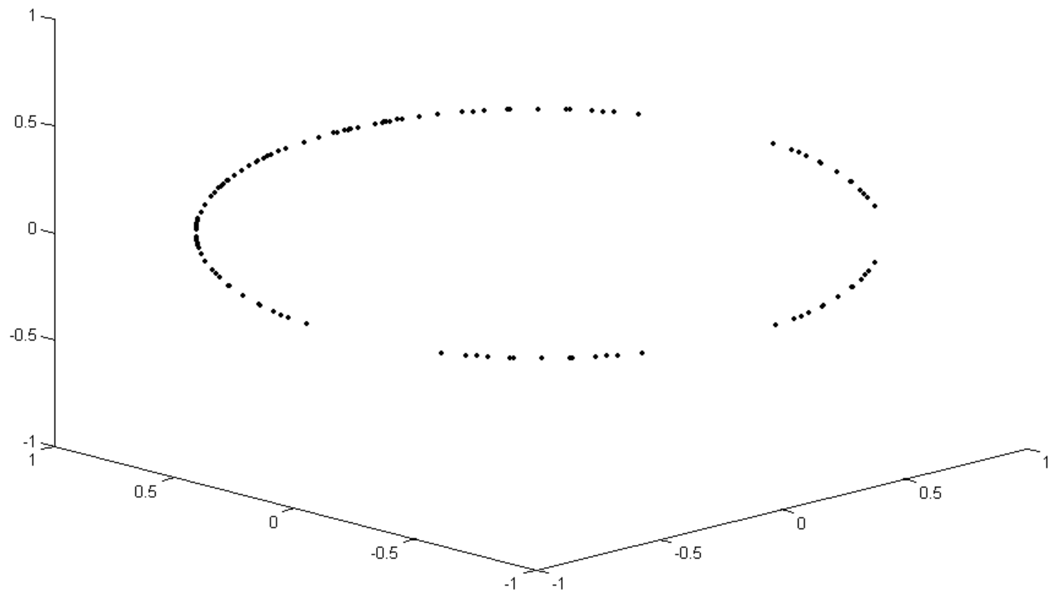


Figure 4.1. This shows the plot of the normalized three-dimensional data.

In addition to providing a data set which represented time-series data, the synthetic four-dimensional data file also provided a sample data set which could be normalized. A plot of the data set was shown in Figure 3.4. Normalizing reduces the dimensionality of the data to three. In order to plot the data in three-dimensional space, principal component analysis is used. I. T. Jolliffe, in the book *Principal Component Analysis*, says,

> the central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set which consists of a large number of interrelated variables, while retaining as much as possible of the variation

> present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables. [14]

The theory and further explanation of principal component analysis can be found in reference [14]. To obtain the principal components for the four-dimensional data, the normalized data was imported into Matlab as a 4×16 matrix, $\mathbf{x}$. The eigenvectors were obtained for the 4×4 matrix, $\mathbf{M} = \mathbf{x} * \mathbf{x}'$ where $\mathbf{x}'$ is the transpose of $\mathbf{x}$. As Jolliffe indicated, the eigenvalues of $\mathbf{M}$ correspond to the variation of the data in the direction of the eigenvectors of $\mathbf{M}$, which are called the principal components. Because the normalized data have dimension $n-1$, at least one of the eigenvalues is zero. The eigenvector matrix, $\mathbf{S}$, consists of the eigenvectors of $\mathbf{M}$ placed in ascending order according to eigenvalues. Multiplying the transpose of the eigenvector matrix by the matrix $\mathbf{x}$, $(\mathbf{S}' * \mathbf{x})$, results in a 4×16 matrix in which the first row is all zeros. By discarding the first row and obtaining a 3×16 principal component matrix, the four-dimensional data has been transformed into three-dimensional data. Figure 4.2 plots the three-dimensional principal component matrix from the four-dimensional data. The four clusters are easily seen. This approach has been used for the data provided with GeneCluster and for the AFRL hydrazine data and will be presented elsewhere [15].

Results from various clustering methods were compared using both the normalized and raw data files of the four-dimensional data. In Table 4.5 the results are displayed as a count of the incorrectly clustered points. Cluster's k-means algorithm is included in this table because it was hypothesized that Cluster would be able to find the true clusters when normalized data was used. In one instance a clustering method did not produce the number of clusters that were desired, in this case, four. The error sum of squares for each set of results were also calculated and are displayed in Table 4.6. Since the genomic and proteomic data are considered similar to the synthetic four-dimensional data set with respect to the higher dimensional time step,

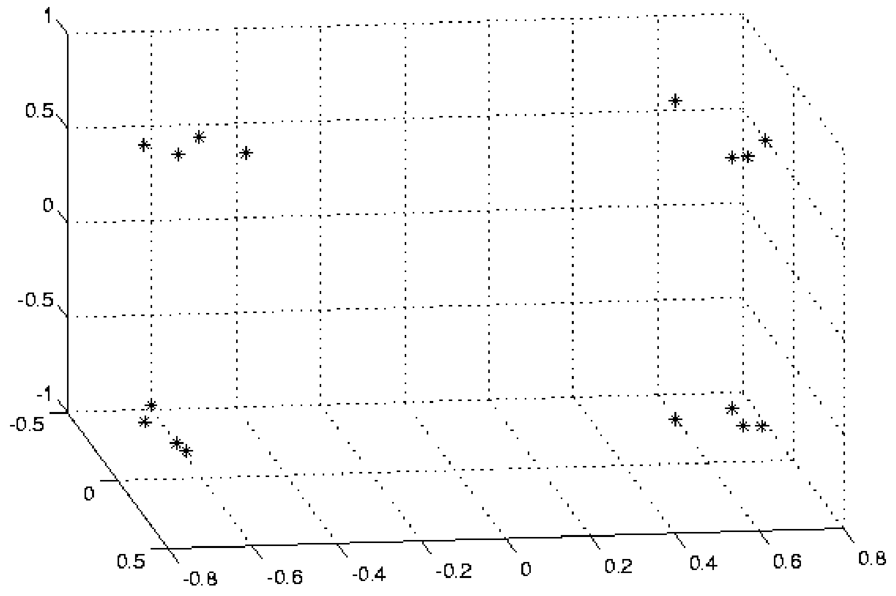


Figure 4.2. The shows the three-dimensional plot of the synthetic, normalized four-dimensional data using three principal components.

Table 4.5. Count of incorrectly clustered points using 4-D synthetic data sets.

Program	Method	4-D	4-D Norm
Matlab	Single	0	0
Matlab	Average	0	0
Matlab	Complete	0	0
Cluster	Single	0	0
Cluster	Average	0	0
Cluster	Complete	0	0
Cluster	K-means	3	3
Cluster	SOM (1×4)	4	1
GeneCluster	SOM (2×2)	8(2)	8(2)
GeneCluster	SOM (4×1)	0	0
SAS	Single	0	0
SAS	Average	0	0
SAS	Complete	0	0
SAS	K-means	0	0

Table 4.6. Error sum of squares for the 4-D synthetic data sets.

Program	Method	4-D	4-D Norm
Matlab	Single	0.7596	0.7596
Matlab	Average	0.7596	0.7596
Matlab	Complete	0.7596	0.7596
Cluster	Single	0.7596	0.7596
Cluster	Average	0.7596	0.7596
Cluster	Complete	0.7596	0.7596
Cluster	K-means	8.7676	8.7676
Cluster	SOM (1×4)	4.3377	2.2086
GeneCluster	SOM (2×2)	9.2092	9.2092
GeneCluster	SOM (4×1)	0.7596	0.7596
SAS	Single	0.7596	0.7596
SAS	Average	0.7596	0.7596
SAS	Complete	0.7596	0.7596
SAS	K-means	0.7596	0.7596

I chose to include several single linkage algorithms. The data set did not prove to be much of a challenge for most of the programs it was tested on. All hierarchical algorithms correctly identified the four true clusters. In SAS, the k-means algorithm also correctly identified the true clusters. Problems began to occur with a specific configuration in GeneCluster and the k-means algorithm in Cluster.

The results from GeneCluster's SOM depended on the configuration of the clusters. When a 4×1 configuration was used, the algorithm correctly identified the four clusters. However using a 2×2 configuration yielded two clusters with eight observations in each and two empty clusters. Each of the resulting clusters contains two of the true clusters. It is interesting to note that all eight observations that have a downward trend between the third and fourth time steps are clustered together. Similarly, all eight observations that have an upward trend between the third and fourth time steps are clustered together. Figure 4.3 shows the plots of these two clusters. A similar result could have been obtained by clustering based on the trend

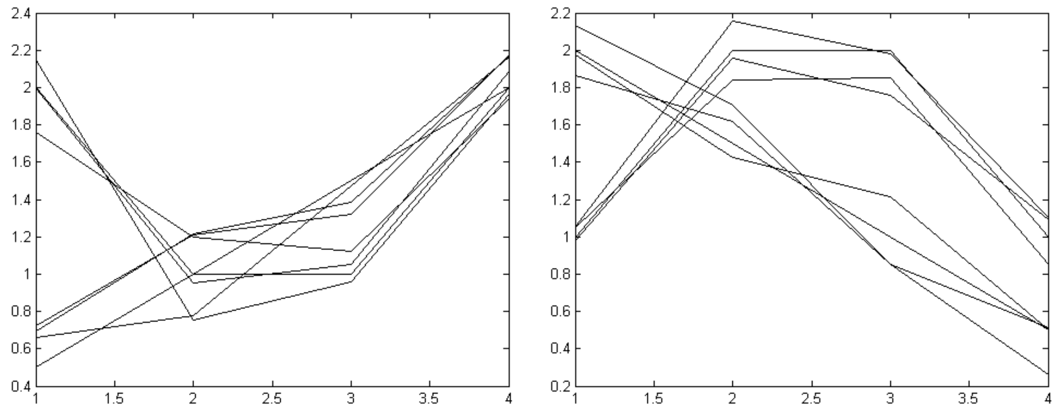


Figure 4.3. Each plot shows the resulting clusters from GeneCluster with the 4-D synthetic data.

between the first and second time step, with the difference being which true clusters are clustered together.

Eisen's k-means algorithm in Cluster incorrectly clustered three data observations. Each of the incorrectly clustered observations appeared in a different cluster, resulting in one cluster of three correct observations, one cluster with all four correct observations and an incorrect observation, and two clusters with three correct observations and one incorrect observation. Cluster allows the input of max cycles which affects the results. At ten max cycles the results are poor. Consequently, this result was obtained using at least one hundred max cycles. With the exception of the k-means algorithm, the Cluster program produced reasonable results for the four-dimensional data set using all of the algorithms. This is an improvement over the two- and three-dimensional data where Cluster did not give reasonable results using any of the algorithms. Hopefully, this indicates that Eisen's Cluster program will work better on the genomic and proteomic data sets that consist of time step data. It still remains unclear how the k-means algorithm in Cluster is implemented, and what kind of data set it will work well with. One assumption is that the algorithm will only work well on a specific kind of data set. If this assumption is true, then since the underlying structure and behavior of the genomic and proteomic data sets are unknown, it would not be feasible to use the k-means algorithm in Cluster. However, if the problem corresponds to how the data is preprocessed and normalized, we could potentially use the algorithm once we know how to cater to the algorithm.

It was previously mentioned in section 3.3 that the data within each data set was ordered in two different ways. By sequentially ordering the data with respect to the true clusters, the results were very easy to interpret. It was hypothesized that since algorithms often work sequentially through the data, a non-sequentially ordered data set may produce different results. Therefore, results were produced using both sequential and non-sequentially ordered data sets. The order of the data within the data set did not make any difference in the clustering results for most of the programs. The only exception is the k-means algorithm in SAS. This method returned more accurate results when a non-sequential order was used. In the previous tables, all of the results were obtained using the sequentially ordered data sets. This was done to show the worst-case results for the SAS k-means algorithm. When better results were obtained with non-sequentially ordered data, usually the difference was very small. However, in two cases it made a significant difference. Using the two-dimensional elliptical data set, B_e2, SAS k-means produced perfect results for the non-sequentially ordered data set, which gives an error sum of squares value of 3.024. Results for the sequentially ordered data set gave an error sum of squares value of 4.357 with sixteen incorrectly clustered points. A significant difference in error sum of squares values was also found when using the three-dimensional data set, $D3$. Perfect results were again obtained when the non-sequentially ordered data were used, giving an error sum of squares value of 19.757. The sequentially ordered data set produced fifteen incorrectly clustered points and an error sum of squares value of 29.248. Since the order of genomic and proteomic data are unknown with respect to how the data should cluster, it is important to realize that some programs, like SAS k-means, can produce varying results.

4.2 GeneCluster Data

The data set provided with the the program GeneCluster, as described in section 3.4, was discussed in the article *Interpreting Patterns of Gene Expression with Self-organizing Maps: Methods and Application to Hematopoietic Differentiation*, [17]. Preprocessing and reducing the data set as explained in the article reduced the data set to less than 600 genes. The reduced data set was then analyzed using various clustering programs. Figure 4.4 shows the GeneCluster output that was obtained from the preprocessed data with a configuration of 4×3. This output corresponds fairly well with the output shown in the article. The article then explains a

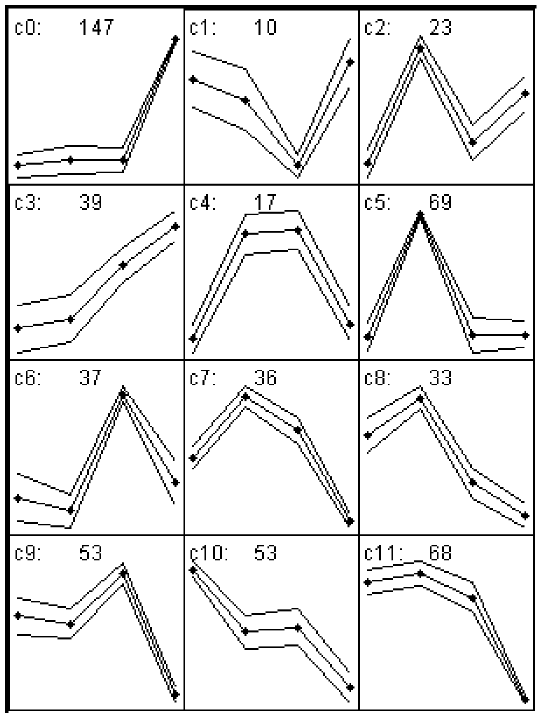


Figure 4.4. This shows the visual output from GeneCluster from the filtered data file that accompanied the software.

specific cluster in detail. Using this specific cluster, Tables 4.7 and 4.8 compare the results from several of the clustering programs. In both tables, the first column gives the gene identifier, and the remaining columns give results from various clustering programs. In order, these programs are GeneCluster (GC), Matlab average linkage (MA), Matlab complete linkage (MC), SAS average linkage (SA), SAS complete linkage (SC), SAS k-means (SK), and Cluster k-means (CK). Table 4.7 includes the genes which are discussed in the article and were clustered together by GeneCluster during this research.

Table 4.7. Cluster results of data set provided by GeneCluster, part 1.

ID	G C	M A	M C	S A	S C	S K	C K
D90144	1	2	1	2	1	1	2
U27467	1	1	1	1	1	1	1
Z11697	1	1	1	1	1	1	1
M60278	1	1	1	1	1	1	1
R37964	1	1	1	1	1	1	1
X51345	1	1	1	2	1	1	2
M31516_i	1	1	1	1	1	1	1
U20158	1	1	1	1	1	1	1
M59465	1	1	1	1	1	1	1
R70479_f	1	1	1	1	1	1	1
T48759	1	1	1	1	1	1	1
X62570	1	1	1	1	1	1	2
X61123	1	1	1	1	1	1	1
M23379	1	2	1	2	1	1	1
Z17227	1	1	1	1	1	1	2
L20859	1	1	1	1	1	1	1
M93425	1	2	1	2	1	1	2
H74178	1	1	1	1	1	1	1
L37042	1	2	1	2	1	1	2
M55268	1	1	1	2	1	1	2
T61599	1	1	1	1	1	1	1
T87873	1	1	1	1	1	1	1
T57483	1	1	1	1	1	1	1
U28918	1	1	1	1	1	1	2
T53118	1	1	2	1	2	1	1

The top section of Table 4.8 shows genes which were mentioned in the article but were not clustered by GeneCluster with the rest of the referenced genes. These are included to show whether or not other clustering programs were able to better reproduce the results provided in the article. The bottom section of Table 4.8 includes genes that were not mentioned in the article but which GeneCluster clustered with the referenced genes. A value of “1” in the tables represents that the gene was placed in the main cluster. Other values represent other clusters. The only two clustering programs which gave the same results were Matlab complete linkage and SAS complete linkage. However, it is clear that all seven programs give good results. The majority of the genes that were not placed in the main cluster were usually clustered together, as indicated by the multiple instances of “2” in each column.

Table 4.8. Cluster results of data set provided by GeneCluster, part 2.

ID	G C	M A	M C	S A	S C	S K	C K
X86809	2	1	1	1	1	1	1
M31516_f	2	1	3	1	3	3	1
M31516_r_i	2	1	1	1	1	3	1
R09561_f	2	1	3	1	3	3	1
J04076	2	1	1	1	1	1	1
H80240_f	2	1	3	1	3	3	1
H80240_i	3	4	4	4	4	4	3
D90145	1	2	1	2	1	1	2
H40677	1	2	1	2	1	5	2
H46624	1	2	1	2	1	1	2
H81068	1	1	1	2	1	1	4
J02685	1	2	1	2	1	1	2
R31698_r	1	3	2	3	2	2	1
R38636	1	2	1	2	1	1	2
R99907_i	1	2	1	2	1	1	2
R99907_r_i	1	3	2	3	2	2	2
T57701	1	3	2	3	2	2	2
U05875	1	2	1	2	1	1	2
U22055	1	3	2	3	2	2	1
U25165	1	1	1	2	1	1	2
X00700_f	1	3	2	3	2	2	1
X02744_r	1	1	2	1	2	1	1
X70991_i	1	2	1	2	1	1	2
cre	1	3	2	3	2	2	1

4.3 Genomic Data

The genomic data set obtained from AFRL was used to begin this research. During the trial and error process with the genomic data set, it was clear that the data needed to be normalized. The genomic data set contained $1,031$ transcripts. Using GeneCluster and specifying twenty-four clusters should result in approximately forty-three genes per cluster. Results from GeneCluster regularly yielded many clusters with only a few genes and often two clusters containing several hundred genes. The error bars for most of the clusters were very large and the centroids seemed to lack any significant shape. Figure 3.7 in section 3.5 shows an example of GeneCluster output from the genomic data. Closer inspection of the large clusters showed that genes within the same cluster varied greatly in behavior. Also, values in the data set varied by tens of thousands. It is desired to have genes clustered together which behave similarly throughout the time course. The actual values of genes that behave similarly may vary by an amplification factor. Normalizing reduces the variance in the data to allow for clustering based on the overall behavior. After normalizing the data, the results from GeneCluster improved greatly. The genes were more evenly distributed to all of the clusters, the error bars were reduced and the centroids showed distinctive shapes, indicating that the genes were clustered according to their overall behavior. Figure 4.5 gives an example of GeneCluster output when the genomic data have been normalized.

Using Matlab, different problems were encountered. Single linkage, the default method, was initially used with the genomic data. Results were obtained using both raw and normalized data. In both instances the same problem was encountered. The dendrogram could not be broken down into a reasonable amount of clusters where each cluster contained a similar number of genes as the next cluster. Figure 4.6 shows a dendrogram from one of the genomic exposure data files. The problem is that instead of producing several distinct clusters, the algorithm chains the genes together. Figure 4.7 is a dendrogram of the top fifty genes from the same genomic

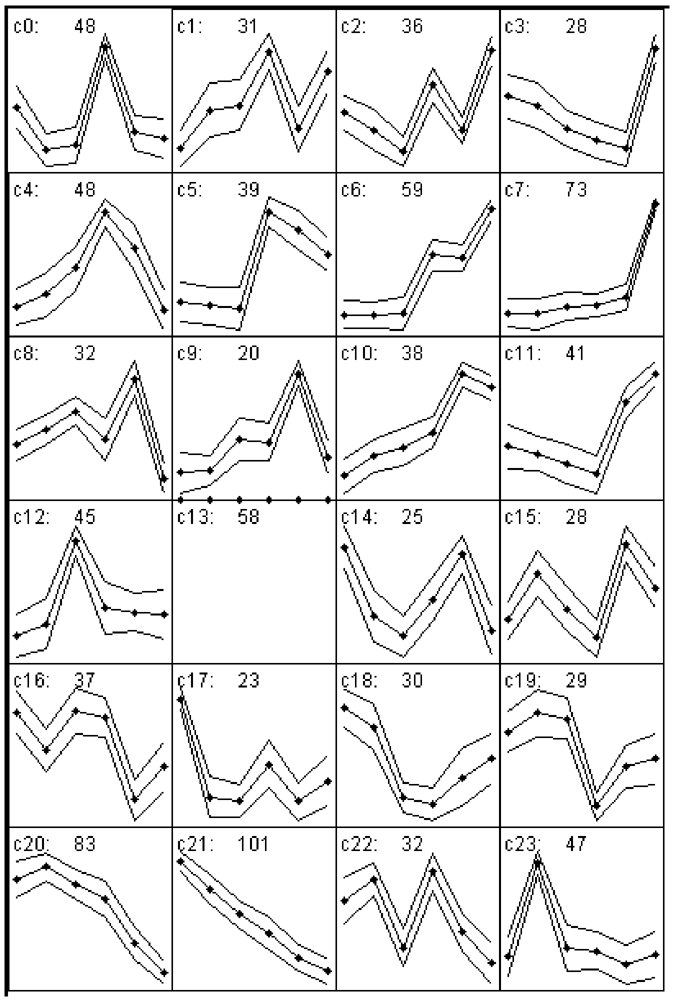


Figure 4.5. This visual output from GeneCluster was obtained using a normalized genomic data file.

exposure file. This close view clearly shows that if two clusters were desired, only one gene would exist in the first cluster while the remaining genes would fall into the second cluster. If more clusters are desired, one cluster will continue to contain most of the genes while the remaining clusters contain only a few genes each.

The chaining tendency of single linkage does not give desirable results in this case. Instead of using single linkage, average and complete linkage were then considered using Matlab. Chaining continued to occur when the data were not normalized. After the data were normalized, the resulting dendrograms from these linkage meth-

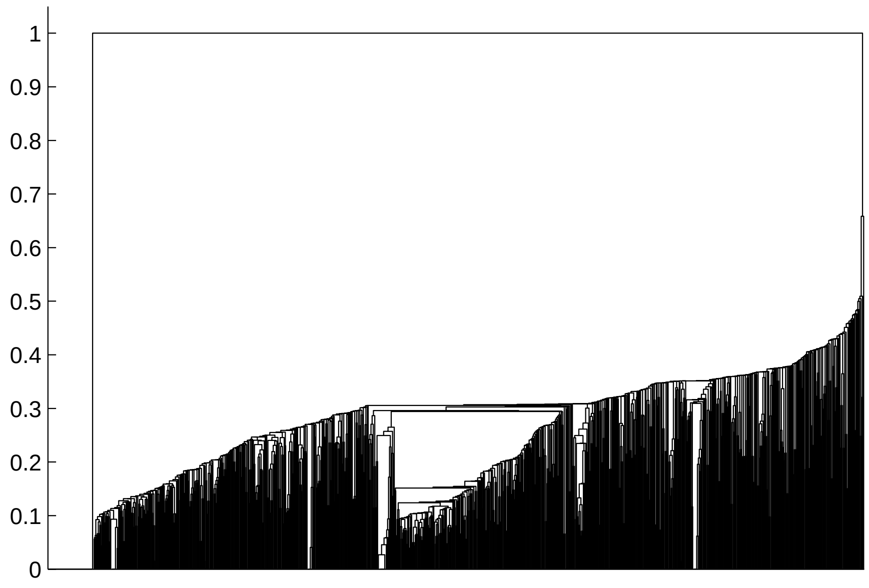


Figure 4.6. This dendrogram shows the chaining tendency of single linkage using a genomic exposure data file.

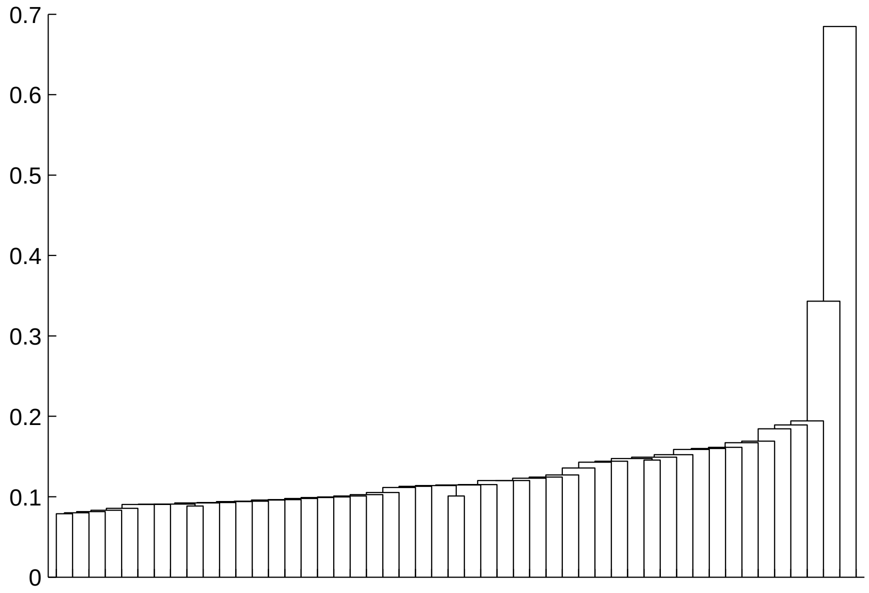


Figure 4.7. This dendrogram shows the top 50 genes of the same genomic exposure data file.

ods improved, as shown by the examples in Figures 4.8 and 4.9. Figure 4.8 is an example of average linkage while Figure 4.9 is the result of complete linkage on the same genomic exposure file. The dashed line indicates where the dendrogram should be cut to produce twenty-four clusters. The actual cluster results are not obtained from visually inspecting the dendrogram. A file can be produced in Matlab that lists the cluster number of each gene. Comparison of the cluster numbers from average and complete linkage shows that although the results of the two methods are not exactly the same, they are much the same, with many of the same genes clustered together.

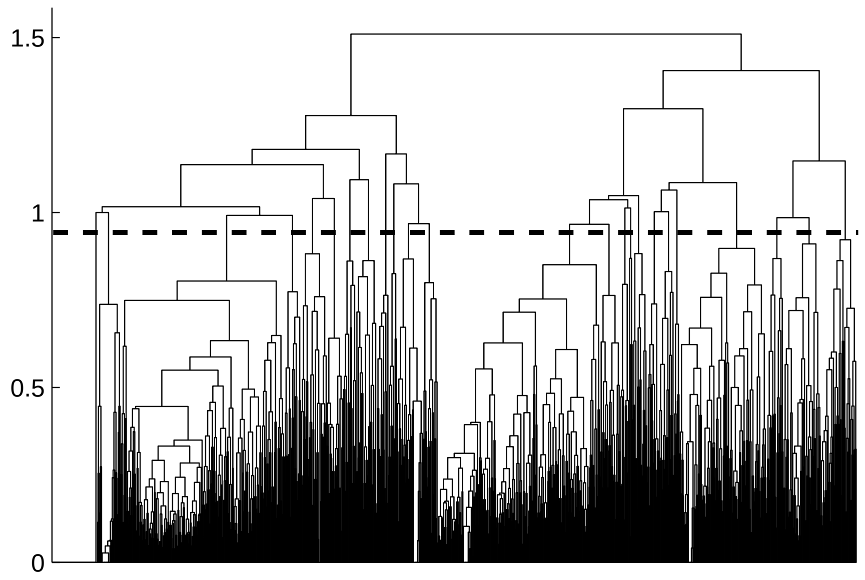


Figure 4.8. This dendrogram shows the results of a genomic exposure data file using average linkage.

4.4 Proteomic Data

The proteomic data offers an opportunity to look at a data set that falls, in size, between the synthetic data and the genomic data. With 110 proteins, the data set is small enough to carefully analyze the results, but large enough to provide a challenge to the clustering programs.

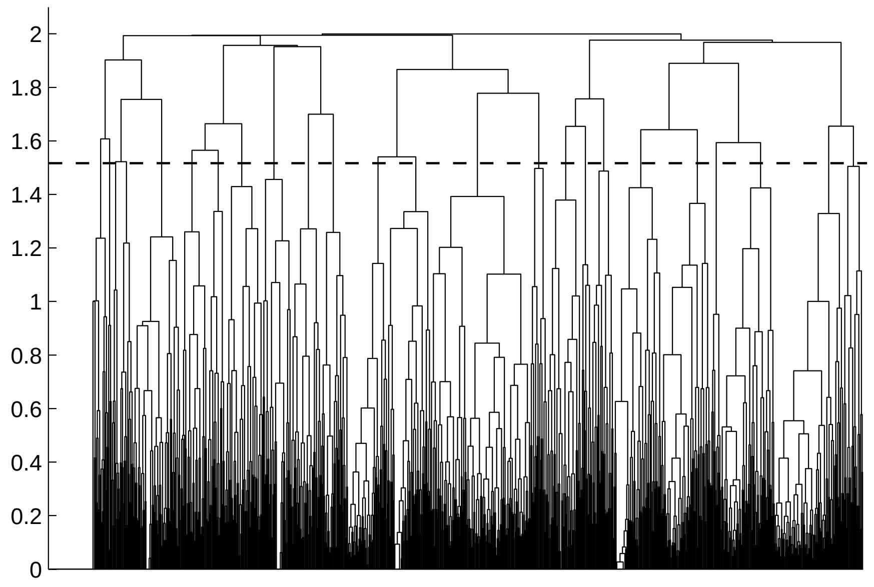


Figure 4.9. This dendrogram shows the results of a genomic exposure data file using complete linkage.

In order to compare the results from several different clustering programs, a file was created with each column containing cluster numbers from a different program. By sorting these columns into their clusters, it is easy to read across the columns to see if proteins were clustered together by more than one program. The file that was most closely examined in this process was the 50 millimole exposure file with normalized data. GeneCluster, Cluster, Matlab, and SAS were all used to cluster the data. The results were the same from complete linkage in both Matlab and SAS. Average linkage from both programs produced results which were very similar. Using $10,000$ epochs in GeneCluster, the proteomic data did not produce consistent results through multiple trials. Table 4.9 gives the error sum of squares for each of the methods using the 50 millimole exposure file with normalized data. The program whose results gave the smallest value for the error sum of squares is Eisen's Cluster program using the k-means algorithm. It is interesting to note that the three programs using a complete linkage algorithm had similar error sum of squares values. The results from GeneCluster and SAS k-means had the highest error sum of squares value.

Table 4.9. Error sum of squares for the protein data set. The values are a factor of 10^9 larger.

Program	Method	ESS
Matlab	Average	8.1760
Matlab	Complete	7.7681
Cluster	Average	8.1454
Cluster	Complete	7.7680
Cluster	K-means	7.5529
Cluster	SOM (1×12)	8.1666
GeneCluster	SOM (3×4)	8.4608
GeneCluster	SOM (6×2)	8.3323
SAS	Average	8.1783
SAS	Complete	7.7681
SAS	K-means	8.4337

4.5 Genomic/Proteomic Data

After collecting results from GeneCluster and the single linkage algorithm in Matlab using the separate genomic and proteomic relational data sets, we wanted to see if the results reflected the relationship between the two sets. In preparation of comparison, we numbered each of the proteins one through sixteen (1–16), and the genes numbered one through twenty-one (1–21), and called this the "relational number." Table 4.10 lists the proteins and genes with the relational numbers and other important identifying information. Table 4.11 shows some of the similar clusterings between the related genes and proteins that Matlab was able to identify. Each section indicates a different subset of the original data. The "E" in the subset title indicates an exposure file and the "T" indicates a time file. The number following either the "E" or "T" indicates which exposure or time step file it is. Following the number is a set of letters that indicate whether the set was normalized ("ns"), or whether it was a difference-normalized file ("Ins"). Under the subheading of "Protein," each line

Table 4.10. Protein and Gene Relational Numbers.

Protein Information			*Gene Information*	
Protein Rel. #	*ID*	*GeneBank*	*Gene Rel. #*	*Affymetrix ID*
1	P00173	AF007107	1	AF007107_s_at
2	P06761	M14050	2	M14050_s_at
3	P06762	J02722	3	J02722cds_at
4	P07632	Y00404	3	Y00404_s_at
4		M25157	5	M25157mRNA_i_at
5	P07901	J04633	6	rc_AA819776_f_at
6	P08010	J03914	7	J03914cds_s_at
7	P08109	M11942	8	M11942_s_at
8	P15651	J05030	9	J05030_at
9	P19226	U68562	10	U68562mRNA#2_s_at
9		X54793	11	X54793_at
10	P23457	D17310	12	D17310_s_at
11	P34067	L17127	13	L17127_at
11		L17127	14	L17127_g_at
12	P49889	M86758	15	M86758_at
12		S76489	16	S76489_s_at
13	P50237	L22339	17	L22339_at
13		L22339	18	L22339_g_at
14	Q63538	X96488	19	X96488cds_at
15	Q63617	U41853	20	U41853_at
16	Q64680	U48220	21	U48220_at

indicates proteins that clustered together and the corresponding genes that clustered together are listed beside them under the subheading of "Gene." More genes may be listed than proteins due to the fact that we had multiple genes relating to one protein. All genes that corresponded to a specific protein were listed if they clustered together.

The results in Table 4.11 are promising. It indicates that some related genes and proteins are behaving similarly at a specific point in the experiment. Finding biological connections between separately clustered data sets shows that clustering techniques are useful for genomic and proteomic research. However, all of the related

Table 4.11. Similar clustering of relational protein and genes using Matlab.

E0ns		*E50ns*	
Protein	*Gene*	*Protein*	*Gene*
14, 16	19, 21	11	13, 14
15	20	9, 3, 15	11, 10, 3, 20
6, 13	7, 17, 18	1, 12	1, 15, 16
4, 12, 8	4, 5, 15, 16, 9	6, 10, 13	7, 12, 17, 18
5, 9, 7	6, 10, 11, 8	2, 5, 7	2, 6, 8
E75ns		*E50Ins*	
Protein	*Gene*	*Protein*	*Gene*
14	19	16	21
15, 3	20, 3	3, 8	3, 9
5, 7	6, 8	6, 14	7, 19
6, 13	7, 17, 18	11, 1	13, 14, 1
1, 12	1, 15, 16	2, 9	2, 10, 11
E75Ins		*T3ns*	
Protein	*Gene*	*Protein*	*Gene*
2, 7, 8	2, 8, 9	14, 11	19, 13, 14
3, 5, 14	3, 6, 19	2, 16, 5, 3	2, 21, 6, 3

proteins and genes did not behave similarly, and those that did behave similarly did not do so throughout the entire experiment.

GeneCluster was also used in attempting to find similar behavior between related proteins and genes. The process of finding such similarities is much more difficult using GeneCluster due to the nature of the output. Instead of producing a comprehensive table of similar clusterings, it was simply verified that the results from GeneCluster were similar to those obtained from Matlab.

V. Summary and Conclusions

5.1 Summary

This research explored the use of several different clustering algorithms to make sense of genomic and proteomic data. Before clustering programs can be used with any data set, the data had to be preprocessed. Several synthetic data sets in two-, three-, and four-dimensions were created in which the true clustering results were known. These data sets were used in order to compare the accuracy of various clustering programs. A genomic data set was obtained with the software GeneCluster. Results from various clustering programs with this data set were compared to results from previous research [17]. Finally, results using the genomic and proteomic data from AFRL were also examined. A relationship was established between several of the genes and proteins. The subset of relational genes and proteins were clustered and examined.

5.2 Conclusions

Preprocessing the data for use with clustering programs is very important. Genomic and proteomic data contain a large amount of variability. By normalizing the data during preprocessing, the variability is reduced and the behavior of the data over time can be examined. Using the synthetic data sets, GeneCluster seemed to consistently give the most reliable results. However, the results were dependent upon the configuration of the clusters. From the data set provided by GeneCluster, all clustering methods seemed to do equally well. Using the proteomic data, Table 4.9 shows that the k-means algorithm from Eisen's Cluster program gave the smallest error sum of squares while results from GeneCluster gave the highest error sum of squares. Despite the differences in the clustering results, all of clustering methods were able to identify data points which should be clustered together. In analyzing genomic and proteomic data, it is best to compare the results from several clustering algorithms.

While comparing clustering algorithms, a genomic data set from previous research was used. Tamayo et al. identified genes within a cluster which had biological relevance [17]. The clustering algorithms used in this research were able to produce similar results with the same data set. Most of the genes detailed by Tamayo were clustered together. These results imply that clustering algorithms are valuable tools for finding biological significance within genomic data.

This research also examined small subsets of the genomic and proteomic data containing related genes and proteins. These subsets were clustered separately. The clusters from the proteomic set were examined and compared to the clusters of the genomic data set. It was observed that when several proteins clustered together, often the genes related to those proteins also clustered together. These results indicate that the biological connection between genes and proteins can be identified using clustering algorithms.

5.3 Future Work

Future research in the area of clustering genomic and proteomic data should further discuss and connect biological meaning to the clustering results. It would be beneficial to find a relationship between genes which behave similarly over time in order to help identify unknown genes. Finding biological meaning within the individual clusters of the related gene and protein data sets could also be beneficial. This would show that cluster analysis can be used to find biological significance in two different ways, and may also help further our understanding of biological pathways. Also, there is concern about the function of genes and proteins after exposure to chemicals like hydrazine. By connecting biology to the clustering results, further research can explore the effects of those chemicals on particular genes or proteins.

Bibliography

1. Alon, U., et al. "Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays," *Proc. Natl. Acad. Sci. USA*, *96*:6745–6750 (June 1999).
2. Anderberg, Michael R. *Cluster Analysis For Applications*. New York: Academic Press, 1973.
3. Backer, Eric. *Computer-assisted Reasoning in Cluster Analysis*. Englewood Cliffs: Prentice Hall, 1995.
4. Chan, Victor, "Microarrays Platform." Presentation, Molecular Bio-Computing Workshop, May 2002.
5. Coleman, Dan, et al. "Some Computational Issues in Cluster Analysis with No Priori Metric," *Computational Statistics & Data Analysis*, *31*:1–11 (1999).
6. Dillon, William R. and Matthew Goldstein. *Multivariate Analysis*. New York: John Wiley & Sons, 1984.
7. Eisen, Michael. *Cluster and TreeView*. Stanford University, http://rana.lbl.gov/, 1998-1999.
8. Everitt, Brian S. *Cluster Analysis* (Third Edition). London: Edward Arnold, 1993.
9. Hartigan, John A. *Clustering Algorithms*. New York: John Wiley & Sons, 1975.
10. Hawkins, Douglas M., editor. *Topics in Applied Multivariate Analysis*. Cambridge: Cambridge University Press, 1982.
11. Hawkins, Douglas M., "Some Ugly Examples of Clustering." Personal Correspondence, 2002.
12. Hill, A. A., et al. "Genomic Analysis of Gene Expression in C. elegans," *Science*, *290*:809–811 (October 2000).
13. Jain, Anil K. and Richard C. Dubes. *Algorithms for Clustering Data*. Englewood Cliffs: Prentice Hall, 1988.
14. Jolliffe, I. T., editor. *Principal Component Analysis*. New York: Springer-Verlag, 1986.
15. Quinn, Dennis W. Personal Communication, September 2002.
16. Swiss Institute of Bioinformatics, "SWISS-PROT Protein Knowledgebase TrEMBL Computer-annotated Supplement to SWISS-PROT." http://us.expasy.org/sprot/, 2002.

17. Tamayo, Pablo, et al. "Interpreting Patterns of Gene Expression with Self-organizing Maps: Methods and Applications to Hematopoietic Differentiation," *Proc. Natl. Acad. Sci. USA*, *96*:2907–2912 (March 1999).

18. Theodoridis, Sergios and Konstantinos Koutroumbas. *Pattern Recognition*. San Diego: Academic Press, 1999.

19. Ward, Jr., Joe H. "Hierarchical Grouping to Optimize an Objective Function," *Journal of the American Statistical Association*, *58*:236–244 (1963).

20. Whitehead Institute, "Whitehead Institute/MIT Center for Genomic Research." http://www-genome.wi.mit.edu, 2002.

21. Witzmann, Frank. Personal Communication, 2001.

22. Zhang, Bin, "Generalized K-Harmonic Means—Dynamic Weighting of Data in Unsupervised Learning." Proceedings, First SIAM International Conference on Data Mining, April 2001.

Vita

Rebecca Olson was born in Columbus, Ohio. During her second year of school she moved with her family to Worthington, Ohio. She spent her eighth grade year in St. Cloud, Florida and returned to Worthington for High School. Rebecca graduated from Worthington Kilbourne High School and the Linworth Alternative Program in 1996. She attended Otterbein College in Westerville, Ohio, majoring in mathematics and minoring in computer science. Rebecca graduated magna cum laude with a Bachelor of Science in 2000.

Rebecca attended the Air Force Institute of Technology (AFIT) at Wright-Patterson Air Force Base, Dayton, Ohio to complete a Master of Science in Applied Mathematics.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 09-09-2002	**2. REPORT TYPE** **Master's Thesis**	**3. DATES COVERED** *(From – To)* Sep 2000 – Sep 2002

4. TITLE AND SUBTITLE COMPARING CLUSTERING ALGORITHMS FOR USE WITH GENOMIC AND PROTEOMIC DATA	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Olson, Rebecca, A., Civilian, AD-1501-21	**5d. PROJECT NUMBER** ENR # 2002-008
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 P Street, Building 640, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GAM/ENC/02S-1
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR, Attn: Dr. Kozumbo, Phone: (703) 696-7720, 801 N. Randolph St. Room 732, Arlington, VA 22203	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Human Genome Project and related projects have resulted in the development of a number of new experimental and analytic tools for use in genomic and proteomic research. In the area of toxicogenomics, researchers are concerned with how genes react to exposure to certain chemicals.

The United States Air Force is interested in the effect of exposure to mission-essential chemicals. Although military personnel may come into contact with chemicals such as hydrazine, risk assessment is usually very limited. On the genomic level, risk assessment is a multi-step and multi-disciplinary process. The process begins with an experiment that exposes cells to the chemical. Data from the experiment are obtained using gene chips. The data can then be analyzed.

This research explores the methods of pre-processing and analyzing data. Several different data sets are used to compare the effectiveness of various clustering algorithms and their implementations. Genomic and proteomic data obtained from a hydrazine exposure experiment are then analyzed. A relationship is established between the genomic and proteomic data sets and is used in further analyses.

15. SUBJECT TERMS

Pattern Recognition, Genome, Toxicology, Principal Component Analysis

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dennis W. Quinn, ENC
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)*
U	U	U	UU	71	(937) 255-3636, ext 4522; e-mail: Dennis.Quinn@afit.edu