2011

Class Discovery and Prediction of Tumor with Microarray Data

Bo Liu
Minnesota State University - Mankato

Class Discovery and Prediction of Tumor with Microarray Data

By

Bo Liu

A Thesis Submitted in Partial Fulfillment of the Requirements for

Master of Science

In

Mathematics and Statistics

Minnesota State University, Mankato

Mankato, Minnesota

Apirl 2011

Class Discovery and Prediction of Tumor with Microarray Data

Bo Liu

This thesis has been examined and approved by the following members of the thesis committee.

Dr. Namyong Lee, Advisor

Dr. Deepak Sanjel

Dr. Fei Yuan

Acknowledgments

It has been a great pleasure for me to study at the Minnesota State University, Mankato in the past two years. I received so much help from the faculty, staff and my friends here, which makes my graduation and this thesis possible.

I would like to firstly convey my deepest appreciation to my advisor and mentor, Dr. Namyong Lee, who gave me unlimited encouragement, guidance and support in my study and research. There is an old Chinese saying, “Once a teacher, always a father.” Dr. Lee gives me generous tolerance and supports in my research and my life, just like what my father did. Some succeeded, some failed, others, I don’t know. But I am so grateful to him for providing me all those opportunities to explore.

I am most grateful to Dr. Deepak Sanjel and Dr. Fei Yuan for accepting my request to be my thesis committee members. I extend my thanks to Dr. Mezbahur Rahman, and Dr. Han Wu for helping me and attending my presentations during my research here.

Another import set of people to mention are my friends I meet here. Ellen Henkelman, Kelly Han, Kristin Brever, Ling Zhang, Mai Nguyen, Peiyu Lee, Susan Chen, Zhe Kong all made my research and life easier and more colorful.

I could not have gotten where I am today without the unlimited love and continuous encouragement from my grandparents, my parents, my aunt, my uncle, my cousin and my boyfriend. I am really blessed to have them.

Abstract

Current microarray technology is able take a single tissue sample to construct an Affymetrix oglionucleotide array containing (estimated) expression levels of thousands of different genes for that tissue. The objective is to develop a more systematic approach to cancer classification based on Affymetrix oglionucleotide microarrays. For this purpose, I studied published colon cancer microarray data. Colon cancer, with 655,000 deaths worldwide per year, has become the fourth most common form of cancer in the United States and the third leading cause of cancer - related death in the Western world.

This research has been focuses in two areas: class discovery, which means using a variety of clustering algorithms to discover clusters among samples and genes; and class prediction that refers to the process of developing a multi-gene predictor of class label for a sample using its gene expression profile. The accuracy of a predictor is also assessed by using it to predict the class of already known samples.

Keywords: Microarray, Data Preprocessing, Clustering, Classification, R

Contents

Chapter 1 Introduction

With the development of discovering the entire human gene map, the main emphasis of research work for life scientists has been pushed forward. Within the 40,000 decoded genes, the functions of majority genes are unclear. The work has the most potential is to identify the function of each gene. Thereby, one could use these materials in medical science to distinguish and treat various diseases. Microarray was born at the right time to uncover most genetic functions in a timely fashion. Actually, the area of human health is the most attractive application of microarrays.

Microarray is a new molecular biological technology which can be used to extract useful information from the resulting datasets with the highest efficiency and in a large scale. In all types of microarrays that have been developed at present, complementary DNA (or cDNA) microarray is the most widely used. Recently introduced experimental techniques based on oligonucleotide or cDNA arrays now allow the expression level of thousands of genes to be monitored in parallel. Microarray technology promises not only to dramatically speed up the experimental work of molecular biologists but also to make possible a whole new experimental approach in molecular biology (Xu et al., 2010).

One typically application of microarray is determining whether a person has a certain disease or not, such as colon cancer, by gene expression analysis. Colon cancer includes cancerous growths in the colon, rectum and appendix. It is very important for doctors to diagnose colon cancer earlier and more precisely, so patients can prevent or get treatment effectively and immediately, which consequently improves the survival rate in colon cancer.

A true clinical dataset is analyzed using a combined microarray technique and some mathematical and statistical approaches. The dataset composes of 62 colon tissue samples which include 40 tumor colon tissue samples and 22 normal colon tissue samples. For each sample we have gene expression intensities for 2000 genes selected from 6817 genes by Alon et al. (1999) according to the highest minimum intensity.

I want to cluster the 62 tissue samples into normal colon tissue and cancerous colon tissue two classes using gene expression values we got from microarray experiments, and produce a classifier which can classify unknown new colon tissue samples to already defined classes. Besides, I want to assess the reliability of the results, in other words, how good the clustering process and the classifier are.

Chapter 2 Method

2.1 Microarrays

A microarray, also called a DNA array or gene chip, is usually a substrate (nylon membrane, glass or plastic) on which one deposits an arrayed series of thousands of microscopic spots of DNA oligonucleotides, called features, each containing picomoles (10^{-12} moles) of single stranded DNA (ssDNA) with various sequences. One will refer to the ssDNA printed on the solid substrate as a probe.

What is deposited on the surface of the array depends on the purpose of the array. Some probes are short sections of a gene, while some are DNA elements that are used to hybridize a cDNA or cRDA. The solution containing the cDNA or cRNA sample is called a target. A target could be generated from a particular biological sample which is being examined. When used in gene expression studies, the DNA target used to hybridize the array is obtained by reverse transcriptase reaction from the mRNA extracted from a tissue sample.

The idea is that the DNA in the solution, that contains sequences complementary to the sequences of the DNA deposited on the substrate, will hybridize to those complementary sequences.

Usually, the target is labeled with a fluorescent dye, a radioactive element, or another method. So the hybridization spot on the substrate can be detected and

quantified easily. If the target is labeled with a dye, then illumination with an appropriate source of light will provide an image of the array of features. The intensity of each spot or the average difference between matches and mismatches can be related to the amount of mRNA present in the tissue. If the target is labeled with a radioactive substance, then the image can be obtained by using a photosensitive device.

Since an array can contain tens of thousands of probes, one can label targets with different dyes and process a multichannel experiment at the same time. Then one could transform the raw microarray data, which are images, into a large number of expression values or gene expression matrices.

Microarray in gene expression studies. It has been shown that microarrays can be used to generate accurate, precise and reliable gene expression data. Microarrays can also be used for purely computational purposes such as DNA computation. When microarrays used for computational purpose, they lose their biological meaning. Using the data generated by microarray to solve computationally intractable problems is very efficient, because of their ability of dealing with high dimensional spaces.

Challenges. There are several challenges when using microarrays in gene expression studies.

Noise. In fact, noise is introduced at each step of various procedures: mRNA preparation, labeling, amplification, pin type, surface chemistry, humidity, target volume, hybridization factors (e.g. time, temperature), dust, scanning, quantification, etc. Due to that much noise, one may get different quantitative values after scanning and image processing steps, even when two experiments are conducted using exactly the same materials and procedures.

The challenge becomes more serious when comparing different tissues or different experiments. Perhaps, one may doubt that the variations of a particular gene are due to the noise rather than a real difference between the different conditions tested. Noise effect is an unavoidable factor. The only way to reduce the noise effect is replication.

This will involve experimental design.

Experimental design. Experimental design could help one to find the reason which changes the output in fact by a series of tests. The tests are different with input variables of a process.

Normalization. Normalization is a method that attempts to remove some of variations from the dataset. For example, variations as different mean intensities caused by different quantities of mRNA, non-linear dye effects, etc.

Large number of genes. In a microarray experiment, one may get information of thousands of genes. Among these genes, part of them is not significant

which means they will not influence the result of our experiments. We could use some technique to find the most significant genes and study on them.

Significance. One crucial question is whether there is significant difference between groups. Some statistic techniques can be used to answer the question.

2.2 Data Preprocessing

The reason for Data preprocessing. Naturally, one should not expect to obtain a good model from a poor dataset. In fact, it is rare that the raw dataset is good and sufficient. Although there is no standard method that can be used to check the quality of the data, some options are available. One way to check the quality is to plot the data and see the graphical representation. Even if the graph looks reasonable, data preprocessing is preferred before the actually analysis of the data.

What is preprocessing. According to Wolfram Mathematica (2011),

"Preprocessing is a transformation, or conditioning, of data designed to make modeling easier and more robust. For example, a known nonlinearity in some given data could be removed by an appropriate transformation, producing data that conforms to a linear model that is easier to work with."

General preprocessing techniques.

Logarithm transformation. Let us use the ratio of corrected intensity of a sample gene and intensity of the reference gene as the relative intensity of each

sample gene. In some sense, relative intensity could reduce the systematic variance of fluorescent dye and scanning between genes.

For example, let us consider two sample genes with corrected intensity values of 100 and 10,000 and the intensity value of the reference gene set at 1,000. If one considers the absolute difference between 1,000 and 100 and 10,000, one would think one gene is affected much more than the other, because

$$10{,}000 - 1{,}000 = 9{,}000 \gg 1{,}000 - 100 = 900.$$

However, from the biological point of view, the change from 1,000 to 100, and from 1,000 to 10,000 are the same, because they both are a 10-fold change; one is increase and the other is decrease. If one uses a logarithm transformation, it is apparent that

$$\log_{10} 1000 = 3$$

$$\log_{10} 100 = 2$$

$$\log_{10} 10{,}000 = 4$$

Indicating the changes is $2 - 3 = -1$ for one gene and $4 - 3 = 1$ for the other gene. The values reflect the fact that two genes change by the same magnitude, but in difference directions.

Therefore, logarithm transformation provides data that are easier to interpret and more meaningful from the biological point of view.

Figure 2.2.1 shows another advantage of logarithm transformation. It makes the distribution symmetrical and mostly normal.

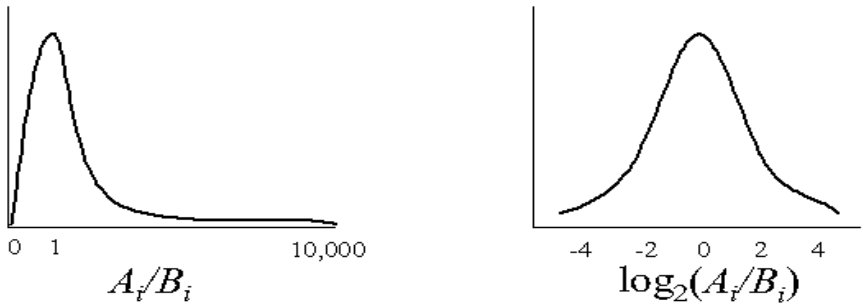

Figure 2.1.1 Distribution before and after logarithm transformation

The left figure shows the distribution of the background corrected intensity values. Note that the intensity range spans a very large interval. The right figure shows the same values after logarithm transformation.

Finally, it is convenient if one uses base 2 logarithms when analyzing the change of gene expressions.

Standardization among microarrays - Median absolute deviation. Each gene chip has hybridized with different sample tissue. Therefore, the first step is adjusting different microarrays to the same level by standardization. After the variables have been standardized, only the general shape of their distributions and the level of their interactions will influence the model. A frequently used method to standardize data is median absolute deviation (MAD). MAD is defined as the median of the absolute deviations from the data's median.

$$MAD = median_i(|X_i - median_j(X_j)|)$$

For example, a univariate dataset $X_1, X_2, \ldots, X_5$ is {2, 2, 3, 4, 14}, and its median is 3. Thus, the absolute deviations from the median are {1, 1, 0, 1, 11}. They can be reordered as {0, 1, 1, 1, 11}. The median absolute deviation is 1.

The advantage of using MAD is that a small number of outliers just slightly influence its value. Whereas, standard deviation (SD) is

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

influenced heavily by outliers, because the distances from the mean are squared, and on average, large deviations are weighted more heavily.

Normalization among paralleled experiments – Quantile Normalization. It is easy to introduce noises into hybridize experiments, so it is easy to produce error. To avoid this, one usually repeats the experiment several times for one sample. Even if one runs the same biological sample in the same experiment twice, one may get slightly different results. To remove some of variations, one could use normalization technique. When normalizing a set of microarrays, there are several approaches which can be used. Quantile normalization is one of them.

The assumption of quantile normalization is that there is an underlying common distribution of intensities across microarrays. The main steps are:

- Let each dataset of a microarray be a column, thus, X with dimension $p * N$ will be a matrix which contains N samples' experiment value with each of them having p elements (p genes).
- Sort each column of X and name it X_{sort}.
- Project each row of X_{sort} onto vector $d = \left(\frac{1}{\sqrt{N}}, \ \ldots\ldots, \frac{1}{\sqrt{N}}\right)$ and get $X_{sort}^{'}$.
- Rearrange each column of $X_{sort}^{'}$ such that every gene returns back to its original place and name it X_{norm}.

2.3 Clustering Analysis of Microarray

After normalizing the microarray data as above, clustering analysis follows. Clustering means group genes by different functions, or similar act of expressions, according to the gene expression data. At the present time, many clustering approaches have been used in gene expression analyses, such as k-means clustering, hierarchical clustering and self-organizing map. In general, all the approaches could be separated into supervised learning and unsupervised learning two categories.

Distance Metric. If one wants to group similar genes together, one should define the meaning of similarity or the measure of similarity first. Such measure of similarity is called a distance or a metric. A distance reflects how close two points are to each other in an input space. The two points can be two genes measured in n different experiments, or two experiments applied on n genes. There are many ways

to calculate the distance and the result of clustering depends on the exact distance metric used.

Properties of Distance metric. A metric or distance function is a function which defines a distance between two points in an n-dimensional space R^n and has the following properties:

- Symmetry. The distance should be symmetric, i.e.:

$$d(x, y) = d(y, x)$$

This means that the distance from x to y should be the same as the distance from y to x.

- Positivity. The distance between any two points should be a real number greater than or equal to zero:

$$d(x, y) \geq 0$$

for any x and y. The equality is true if and only if $x = y$, i.e. $d(x, y) = 0$.

- Triangle inequality.

The distance between two points x and y should be shorter than or equal to the sum of the distance from x to a third points z and from z to y:

$$d(x, y) \leq d(x, z) + d(z, y)$$

This property reflects the fact that the distance between two points should be measured along the shortest route (Drăghici, 2003).

Many different distance metrics can be defined, but they all have the three properties listed above. Furthermore, in clustering and classification problems, the result may vary differently and be affected by the distance metric one used.

Next, I will introduce two different distances.

Euclidean distance. In mathematics, the Euclidean distance is the "ordinary" distance between two points that one would measure with a ruler, and is given by the Pythagorean formula. The Euclidean distance between two n-dimensional vectors $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ is

$$d_E(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

(Drăghici, 2003).

This distance between x and y is the length of the line segment $\overline{xy}$. Thus, in one dimension, the distance between two points on the real line is the absolute value of their numerical difference. Therefore, if x and y are two points on the real line, then the distance between them is computed as

$$d(x, y) = \sqrt{(x - y)^2} = |x - y|.$$

Correlation distance. The Pearson correlation distance between two n-dimensional vectors $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ is

$$d_R(x, y) = 1 - r_{xy}$$

where r_{xy} is the Pearson correlation coefficient of the vectors x and y

$$r_{xy} = \frac{s_{xy}}{\sqrt{s_x}\sqrt{s_y}} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Note that since the Pearson correlation coefficient r_{xy} varies only between -1 and 1, the distance $1 - r_{xy}$ will take values between 0 and 2 (Drăghici, 2003).

The Pearson correlation coefficient r_{xy} reflects the degree of linear relationship between two variables. A correlation of $+1$ means that there is a perfect positive linear relationship between variables, and a correlation of -1 means that there is a perfect negative linear relationship between variables. The Pearson correlation distance $1 - r_{xy}$ focuses on whether the coordinates of the two points change in the same way (e.g. corresponding coordinate increase or decrease at the same time) (Drăghici, 2003). For instance, if $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ represent the measured values of n genes in two different experiments, the Pearson correlation distance will be low if the genes vary in a similar way in the two experiments. These genes would cluster together with the Pearson correlation distance. On the other hand, the Pearson correlation distance close to 2 implies the coordinate for the vector $x = (x_1, x_2, \dots, x_n)$ is increasing (or decreasing) while the corresponding coordinate for the other vector $y = (y_1, y_2, \dots, y_n)$ is decreasing (or increasing). In other words, $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ vary in an opposite way. Therefore, these genes would be grouped into remote clusters.

Let us consider there are three genes have been tested in 5 experiments, and their experiment values are expressed as $g_1 = (1,2,3,4,5)$, $g_2 = (100,200,300,400,500)$ and $g_3 = (5,4,3,2,1)$, respectively. Then, one would group $g_1 = (1,2,3,4,5)$, $g_2 = (100,200,300,400,500)$ in a same cluster, rather than $g_3 = (5,4,3,2,1)$, using the Pearson correlation distance. g_1 and g_2 have a high correlation ($d(g_1, g_2) = 1 - r = 1 - 1 = 0$) whereas g_1 and g_3 are anti-correlated ($d(g_1, g_3) = 1 - r = 1 - (-1) = 2$). However, the Euclidean distance will group g_1 and g_3 in a same cluster, rather than g_2, because $d_E(g_1, g_3) \approx 6.32$ while $d_E(g_1, g_2) \approx 734.20$.

Clustering algorithms. There are several clustering approaches and algorithms. Any of these clustering algorithms can be used to group genes or experiments. The data is usually described by an n-dimensional vector, which is referred to as a pattern or an instance. A set of clusters including all genes or experiments considered form a clustering, cluster tree, or dendrogram.

In most cases, the result of clustering is highly dependent on the distance metric one used. For example, for the same patterns, Euclidean distance metric may produce a different clustering result than the Pearson correlation distance. Furthermore, the same clustering algorithm applied to the same data may produce different results. This is caused by the initialization process, such as a random choice of the initial cluster centers or a random choice of patterns to be used as initial clusters.

As a result, one should always check whether the gene is grouped in the same cluster or not when the clustering algorithm is applied many times. In addition, one must note that in most clustering algorithms (e.g. k-means and hierarchical clustering) the position of the patterns within the clusters does not reflect their relationship in the input space (Drăghici, 2003).

Inter-cluster distances and algorithm complexity. This section will discuss the main methods used to calculate the distance between clusters.

Single linkage. Single linkage method calculates the distance between clusters as the distance between the closest neighbors. It measures the distance between each member of one cluster to each member of the other cluster and takes the minimum of these distances.

Complete linkage. Complete linkage calculates the distance between the furthest neighbors. It takes the maximum of distance measures between each member of one cluster to each member of the other cluster.

Centroid linkage. Centroid linkage defines the distance between two clusters as the squared Euclidean distance between their centroids or means. This method tends to be more robust to outliers than other methods. The centroid of a group of patterns is the point that has each coordinate equal to the mean of the corresponding

coordinates of the given patterns. For instance, the set of experiments: $e_1 = (1,2,3), e_2 = (2,3,4)$ and $e_3 = (3,4,5)$ has the centroid at

$$\left(\frac{1+2+3}{3}, \frac{2+3+4}{3}, \frac{3+4+5}{3}\right) = (2,3,4)$$ (Drăghici, 2003).

Average linkage. Average linkage measures the average distance between each member of one cluster to each member of the other cluster (Korol, 2003, p.23).

Algorithm complexity. The complexity of the algorithm depends very much on which linkage method to be used, as well as its speed. Single or complete linkages require only choosing one of the distances already calculated while more elaborated linkages, such as centroid, require more computations (Drăghici, 2003). However, simple and fast method such as single linkage tends to produce stringy clusters which is bad. While complex and slow method such as centroid linkage tends to produce better clustering which reflect more accurately of the structure of the dataset.

K-means clustering. The k-means algorithm is one of the simplest and fastest clustering algorithms. In consequence, it is also one of the most widely used algorithms. K-means clustering groups patterns into k clusters. Sometimes k is unknown in advance, thus one need to pick a number for k, and then start the algorithm.

The program starts by randomly choosing k points in the same input space as k initial centers of the clusters. These points may be just random points in the

input space, random points from more densely populated volumes of the input space or just randomly chosen patterns from the data itself (Drăghici, 2003). Once the k initial centers have been chosen, the distance from each pattern to every center of the cluster would be computed in a distance metric and that pattern should be associated with the closest cluster center. A first approximate clustering is obtained.

The second step starts by recalculating the new center for every cluster (the center is calculated as the centroid of the group of patterns). Since the centers have moved, one need to calculate new distance from each pattern to every updated center, and then associates the pattern with the closest cluster center.

The program repeats the second step until the cluster centers are such that no pattern moves from one cluster to another. Since no pattern has changed membership, the centers will remain the same and the algorithm can terminate.

Cluster quality assessment. The results of the k-means algorithm may different if one applies the algorithm again, because the initial cluster centers are chosen randomly. Therefore, one needs to assess the quality of the obtained clustering after every successive run.

One way to assess the goodness of fit of a given clustering is to compare the size of the clusters (d) versus the distance to the nearest cluster (D). If the inter-cluster

distance (D) is much greater than d, the clustering is considered as a good one. Therefore, the ratio of D and d can be used as an indication of the cluster quality.

Another way is to measure the average of the distances between the members of a cluster and the cluster center. Smaller average distances are better than the larger ones because they reflect more uniformity in the results (Korol, 2003, p.25).

The diameter of the smallest sphere including all members of a given cluster may also be used as an indication of the cluster quality. However, this measure is sensitive to cluster outliers, because the diameter of the smallest sphere including all members of the cluster is determined by the furthest pattern from the cluster.

In fact, the same clustering algorithm applied to the same data may produce different results. So, one may interested in whether a gene would fall into the same cluster or not if the clustering algorithm is repeated. This question can be addressed by repeating the clustering several times and following the particular gene of interest. Those genes that are clustered together repeatedly are more likely to be genuinely similar.

K-medoids clustering and PAM algorithm. Both the k-means and k-medoids clustering are breaking dataset up into groups and both attempt to minimize squared error, which is the distance between points in a cluster and the center of that cluster.

PAM algorithm. The objective of k-medoids clustering is to minimize a sum of dissimilarities. Compared to k-means clustering, which tries to minimize a sum of squared dissimilarities, k-medoids clustering is more resistant to noises and outliers. Among many algorithms for k-medoids clustering, Partitioning Around Medoids (PAM) is known to be the most powerful one.

The PAM algorithm is described as follows:

- Randomly select k patterns as the initial medoids. Medoid can be defined as that a pattern of a cluster, whose average dissimilarity to all the patterns in the cluster is minimal.
- Associate each pattern to the closest medoid.
- For each medoid m_i, swap m_i and each non-medoid pattern O, and calculate the sum of dissimilarities.
- The one has the lowest sum of dissimilarities will be updated as new mediod.
- Repeat from step 2 to 4 until there is no smaller sum of dissimilarities.

Hierarchical clustering. Hierarchical clustering produces a tree with leaves and root, where leaves as individual patterns and root as the convergence point of all branches. The diagrams produced by the hierarchical clustering are known as dendrograms.

K-means clustering gives us a set of k clusters. In any given cluster, all the members are on the same level. No particular inferences can be made about the relationship between members of a given cluster or between clusters (Drăghici, 2003). However, in hierarchical clustering, different genes and/or experiments are grouped together to form clusters using a chosen distance metric, and then different clusters are also linked together to form a higher level cluster using one of the inter-cluster distances. Therefore, a dendrogram represents not only clusters but also the relations between the clusters.

A bottom-up algorithm will be used in this research.

The bottom-up algorithm. The bottom-up method starts from the individual patterns (leaves) and works upwards the root. This approach is sometimes called agglomerative because it links similar small clusters to form larger clusters. The bottom-up method is described as follows:

- Each single pattern represents a cluster. The pattern can either be a gene or an experiment depending on what the algorithm is applied to.
- Calculate a table containing the distances from each cluster to other clusters.
- Merge the two most similar clusters into a single super-cluster.
- Repeat step 2 and 3 until the entire tree is constructed.

For a given dataset, using a chosen distance metric, one hierarchical clustering algorithm should always produce the same clustering tree.

Summary. Let us close this section by a few conclusions. First of all, for a given dataset, using the same distance metric, different approaches can produce different clustering trees. Second, one should not judge an algorithm by its speed. Compare to the slow algorithm, faster algorithm may reflect less information of the structure of a given dataset. And the key is to obtain a clustering that reflects the structure of the original dataset. Finally, the place of genes in a dendrogram does not necessarily convey useful information and can be misunderstood. Two genes proximity in a hierarchical clustering does not necessarily correspond to similarity.

2.4 Classification

In machine learning and pattern recognition, classification refers to an algorithmic procedure for assigning a given piece of input data into one of a given number of clusters. An algorithm that implements classification is known as a classifier. Sometimes, classifier also refers to the mathematical function, implemented by a classification algorithm that maps input data to a cluster.

Generally, classification is a supervised procedure, which means the algorithm learns how to classify a test point into one of exist clusters based on what it learned from a training dataset. A training dataset is a set of labeled objects and its information will be used to classify the test points. When one wants to make a

prediction on which cluster a biological sample belongs to by its gene expression data, classification comes in.

There are many possible techniques for data classification. Three classification methods will be introduced, which are linear discriminant analysis, minimum sum of squared-error and perceptron algorithm.

Linear discriminant analysis. Linear discriminant analysis (LDA) and the related Fisher's linear discriminant are methods used in statistics, pattern recognition, and machine learning process to find a linear combination of features which characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier.

LDA does not change the location of original data but only tries to optimize class separability by maximizing the ratio of between-class variance to the within-class variance in any particular dataset. Thereby, the original data will be transformed and a test points can be classified in the transformed space. This method also helps one to get better understanding of the distribution of the original data.

Compared with the unsupervised technique of "Principal Components Analysis (PCA)", which projects data in the directions of maximum variance, Fisher LDA projects patterns to a line such that samples from different classes are well separated. Fisher LDA finds a direction to project data onto it in order to minimize the within-class variance and maximize the between-class variance. Therefore, the

patterns in one class are set to be as close together as possible and each class is as far as possible from each other. To achieve the goal, Fisher LDA considers maximizing the following objective function

$$J(a) = \frac{a^T S_B a}{a^T S_W a}$$

where $S_B = \sum_{i=1}^{c}(\mu_i - \bar{\mu})(\mu_i - \bar{\mu})^T$ is the between classes scatter matrix and $S_W = \sum_{i=1}^{c}\sum_{j=1}^{M_i}(x_{ij} - \mu_i)(x_{ij} - \mu_i)^T$ is the within classes scatter matrix, where c is the total number of classes, M_i is the number of patterns is class i, $\mu_i = \frac{1}{M_i}\sum_{j=1}^{M_i} x_{ij}$ is the mean of class i, $\bar{\mu} = \frac{1}{\sum M_i}\sum_{i=1}^{c}\sum_{j=1}^{M_i} x_{ij}$ is the overall mean of the patterns.

A two-class problem will be illustrated and help to understand the classification method.

Suppose we have two classes and d-dimensional samples $x_1, x_2, \dots, x_M$ where M_1 samples come from the first class, and M_2 samples come from the second class. Vector a in the objective function gives the line direction. Project x_i onto a line in the direction of vector a, then scalar $a^T x_i$ gives not only the distance of projection of x_i from the origin, but also the projection of sample x_i onto the line in direction a. Let $\widetilde{\mu_1} = \frac{1}{M_i}\sum_{i=1}^{M_i} a^T x_i = a^T\left(\frac{1}{M_i}\sum_{i=1}^{M_i} x_i\right) = a^T\mu_1$ be the mean of projections of class 1, and similarly, $\widetilde{\mu_2} = a^T\mu_2$ be the mean of projections of class 2. And $y_i = a^T x_i$ is the projected sample of x_i. Then define a scatter for projected samples of class 1 as $\widetilde{S_1}^2 = \sum_{y_{i\in class\,1}}(y_i - \widetilde{\mu_1})^2$, and define a scatter for projected

samples of class 2 as $\widetilde{S_2}^2 = \sum_{y_i \in class\ 2}(y_i - \widetilde{\mu_2})^2$. Small $\widetilde{S_1}^2$ and $\widetilde{S_2}^2$ imply that projected samples of class i are clustered around projected mean $\widetilde{\mu_1}$ and $\widetilde{\mu_2}$. As a result, a vector a which makes $\frac{(\widetilde{\mu_1} - \widetilde{\mu_2})^2}{\widetilde{S_1}^2 + \widetilde{S_2}^2}$ large will guarante that the classes are well separated. Eventaully, $\frac{(\widetilde{\mu_1} - \widetilde{\mu_2})^2}{\widetilde{S_1}^2 + \widetilde{S_2}^2}$ can be transformed and written as the objective function

$$J(a) = \frac{a^T S_B a}{a^T S_W a}$$

To maximize $J(a)$, take the derivative with respect to a and set it to 0. It is equivalent to solve

$$S_B a - \frac{a^T S_B a}{a^T S_W a} * S_W a = 0 \qquad let\ \frac{a^T S_B a}{a^T S_W a} = \lambda$$

$$\Rightarrow S_B a = \lambda S_W a$$

So, this is a solving generalized eigenvalue problem. If S_W has full rank (the inverse exists), problem can be converted to a standard eigenvalue problem

$$S_W{}^{-1} S_B a = \lambda a$$

Therefore, $a = S_W{}^{-1}(\mu_1 - \mu_2)$, where $S_W = \sum_{i=1}^{c} \sum_{j=1}^{M_i} (x_{ij} - \mu_i)(x_{ij} - \mu_i)^T$.

The following is a concrete example.

Class 1 has 5 samples $c_1 = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 3 \\ 4 & 5 \\ 5 & 5 \end{bmatrix}$ and class 2 has 6 samples $c_2 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 3 & 1 \\ 3 & 2 \\ 5 & 3 \\ 6 & 5 \end{bmatrix}$.

Mean for each class are $\mu_1 = [3\ \ 3.6]$ and $\mu_2 = \left[\frac{10}{3}\ \ 2\right]$. The within class scatter

matrix is $S_W = \begin{bmatrix} -2 \\ -1.6 \end{bmatrix}[-2 \ -1.6] + \cdots + \begin{bmatrix} 8/3 \\ 3 \end{bmatrix}[8/3 \ \ 3] = \begin{bmatrix} 82/3 & 24 \\ 24 & 23.2 \end{bmatrix}$. Since S_W has full rank, so the inverse of S_W is ${S_W}^{-1} = \begin{bmatrix} 0.399 & -0.413 \\ -0.413 & 0.470 \end{bmatrix}$. The optimal line direction is computed by $a = {S_W}^{-1}(\mu_1 - \mu_2) = \begin{bmatrix} -0.794 \\ 0.890 \end{bmatrix}$. As long as the line has the right direction, its exact position does not matter.

A complication in applying LDA is, in many cases, linear discriminant is not suitable. Then, LDA can be extended for use in non-linear classification via the kernel trick. This works as effectively mapping the original data into a higher dimensional non-linear space, and then, implementing linear discriminant analysis in this non-linear space. Thus, using linear classification technique in a non-linear space is equivalent to applying non-linear classification in the original space.

Linear discriminant functions. Assume that one knows the proper forms of the discriminant functions, and then the value of the parameters of the classifier can be estimated by using the samples. Various procedures of determining discriminant functions has been developed, none of them requires knowing the forms of underlying probability distributions.

Linear discriminant functions have a variety of pleasant analytical properties. They can be optimal if the underlying distributions are cooperative, such as Gaussians having equal covariance, as might be obtained through an intelligent choice of feature detectors. Even when they are not optimal, we might be willing to sacrifice some performance in order to gain the advantage of their simplicity. Linear discriminant

functions are relatively easy to compute and in the absence of information suggesting otherwise, linear classifiers are attractive candidates for initial, trial classifiers (Duda, 2001).

Linear dicriminant functions in two-category case. A discriminant function that is a linear combination of the components of x can be written as

$$g(x) = w^T x + w_0$$

where w is the *weight vector* and w_0 the *threshold weight*.

$g(x)$ as a two-category classifier implements the following decision rule

$$\begin{cases} Decide\ x\ to\ class\ 1\ w_1\ \ if\ \ g(x) > 0 \\ Decide\ x\ to\ class\ 2\ \ w_2\ \ if\ \ g(x) < 0 \\ Either\ class\ or\ undefined\ if\ g(x) = 0 \end{cases}$$

The equation $g(x) = 0$ defines the decision surface that separates points assigned to w_1 from points assigned to w_2. When $\mathrm{g}(x)$ is linear, this decision surface is a hyperplane. It is easy to show that vector w is normal to any vector lying in the hyperplane. The normal vector, often simply called the "normal", to a surface is a vector perpendicular to it. Thus, the weight vector w is perpendicular to the decision surface. The hyperplane H divides the feature space into two half-spaces, region R_1 for w_1 and region R_2 for w_2. It is sometimes said that any x in R_1 is on the positive side of H, any x in R_2 is on the negative side.

The discriminant function $\mathrm{g}(x)$ gives an algebraic measure of the distance from x to the hyperplane: $r = \frac{g(x)}{\|w\|}$. In particular, the distance from origin to H is given by

$$r = \frac{g(0)}{\|w\|} = \frac{w^T 0 + w_0}{\|w\|} = \frac{w_0}{\|w\|}$$

The origin is on the positive side, when $w_0 > 0$. The origin is on the negative side, when $w_0 < 0$. The hyperplane H passes through the origin, when $w_0 = 0$. To sum up, the orientation of the surface is determined by the normal vector w, and the location of the surface is determined by the threshold weight w_0. A geometric illustration of these algebraic results is given in Figure 2.2.1.

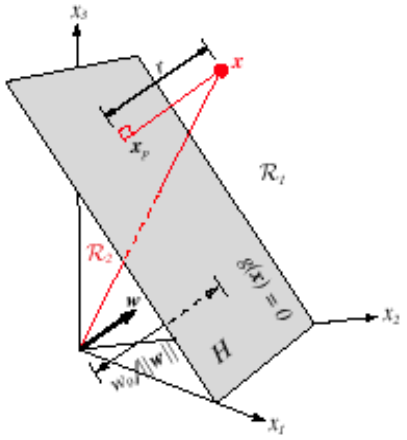

Figure 2.4.1 Geometric illustration of algebraic results

The linear decision boundary H, where $g(x) = w^T x + w_0 = 0$, separates the feature space into two half-spaces R_1 and R_2. In convenient, $g(x) = w^T x + w_0 = 0$ could be written as $g(x) = a^T y$. The reason is

$$g(x) = w_0 + w^T x = w_0 + \sum_{i=1}^{d} w_i x_i = \sum_{i=0}^{d} w_i x_i$$

where $x_0 = 1$. Thus, we can let

$$y = (1, x_1, \ldots, x_d)^T = \begin{pmatrix} 1 \\ X \end{pmatrix}, \text{ and } a = (w_0, w_1, \ldots, w_d)^T = \begin{pmatrix} w_0 \\ W \end{pmatrix}.$$

Regard a two-category case as example, suppose we have a set of n samples $y_1, \ldots, y_n$, some labeled w_1 and some labeled w_2. One major concern is to compute the unknown parameters w_i, $i = 0 \ldots d$, defining the decision hyperplane. A sample y_i is classified correctly if

$$\begin{cases} a^T y_i > 0 \; and \; y_i \; is \; labeled \; w_1 \\ a^T y_i < 0 \; and \; y_i \; is \; labeled \; w_2 \end{cases}.$$

To simplify the method in two-category case, replace all the samples labeled w_2 by their negatives. Therefore, we could look for a vector a such that $a^T y_i > 0$ for all the samples, and such a vector a is called a *separating vector* or a *solution vector*.

Each sample y_i places a constraint, $a^T y_i > 0$, so the solution vector must be on the positive side of every hyperplane $a^T y_i = 0$. The equation $a^T y_i = 0$ defines a hyperplane through the origin of weight space having y_i as a normal vector. Therefore, the intersection of n half-spaces is the solution region. The solution vector should be a vector in this region.

Thus, the problem of finding a linear discriminant function will be formulated as a problem of minimizing a criterion function.

Perceptron algorithm. Now we need to adopt an appropriate criterion function $J(a)$ and an algorithmic scheme to optimize it, and also solve the linear inequalities $a^T y_i > 0$. We choose the *perceptron criterion function* defined as

$$J(a) = \sum_{y \in Y} (-a^T y)$$

where Y is the set of training samples misclassified by a. Obviously, $J(a)$ is always positive and when it takes its minimum value, 0, a solution has been obtained.

An iterative scheme - gradient descent method is adopted to approach the optimization.

$$\mathrm{a(k+1) = a(k) - \eta(k) \nabla J\big(a(k)\big)}$$

where $\eta(k)$ is a positive scale factor or learning rate that set the step size, $\nabla J\big(a(k)\big)$ is the gradient vector $\left(\frac{\partial J}{\partial a_0}, \ldots \frac{\partial J}{\partial a_d}\right)$.

Hence, the preceding rule becomes

$$a(k+1) = a(k) - \eta(k) \sum_{y \in Y} (-y)$$

The algorithm is initialized from an arbitrary weight vector $a(0)$. The weight vector is corrected according to the preceding rule. This is repeated until all features are correctly classified. $\eta(k)$ can be properly chosen as $\eta(k) = \frac{c}{k}$, where c is a constant. The proper choice of the sequence $\eta(k)$ is vital for the convergence speed of the algorithm.

Minimum sum of squared-error. Rather than the perceptron algorithm which only focuses on misclassified training samples, minimum squared-error procedure involves all of the samples. Minimum squared-error tries to find vector a

such that $a^T y_i = b_i$ and also minimizing the sum of squared-error $J(a) = \|Ya - b\|^2 = \sum_{i=1}^{n}(a^T y_i - b_i)^2$.

Let $e_i = a^T y_i - b_i = y_i^T a - b_i$, for each i, so,

$$J(a) = \sum_{i=1}^{N}(b_i - y_i^T a)^2 = \sum_{i=1}^{N} {e_i}^2.$$

To minimize $J(a)$, we could take the derivative with respect to a and set it equals 0.

$$\nabla J(a) = \nabla[\sum_{i=1}^{N}(b_i - y_i^T a)^2] = \sum_{i=1}^{N} 2\,(b_i - y_i^T a) * (-y_i) = 0$$

$$\Rightarrow \left(\sum_{i=1}^{N} y_i y_i^T\right) a = \sum_{i=1}^{N} y_i b_i$$

$$\Rightarrow (Y^T Y) a = Y^T b$$

$$\Rightarrow a = (Y^T Y)^{-1} Y^T b$$

In our research, $b_i = 1$ if the i^{th} sample is known as normal tissue sample and $b_i = -1$ if the i^{th} sample is known as tumor tissue sample.

The advantage of minimum sum of squared-error is that the solution always exists.

Chapter 3 Case Study on Colon Cancer Data and Discussion

3.1 Boxplots of Preprocessed Data

Our dataset consists of 62 colon tissue samples, for each sample we have gene expression intensities for 2000 genes. These 2000 were selected from 6817 by Alon et al. (1999) according to the highest minimum intensity. Within the samples are some paired data. That is, we have a normal and a cancerous tissue sample from the same patient.

In our research, three different data preprocessing are performed on the original 2000 by 62 data. They are log-transformation, standardization, and quantile normalization.

Figure 3.1.1 shows boxplot of original dataset.

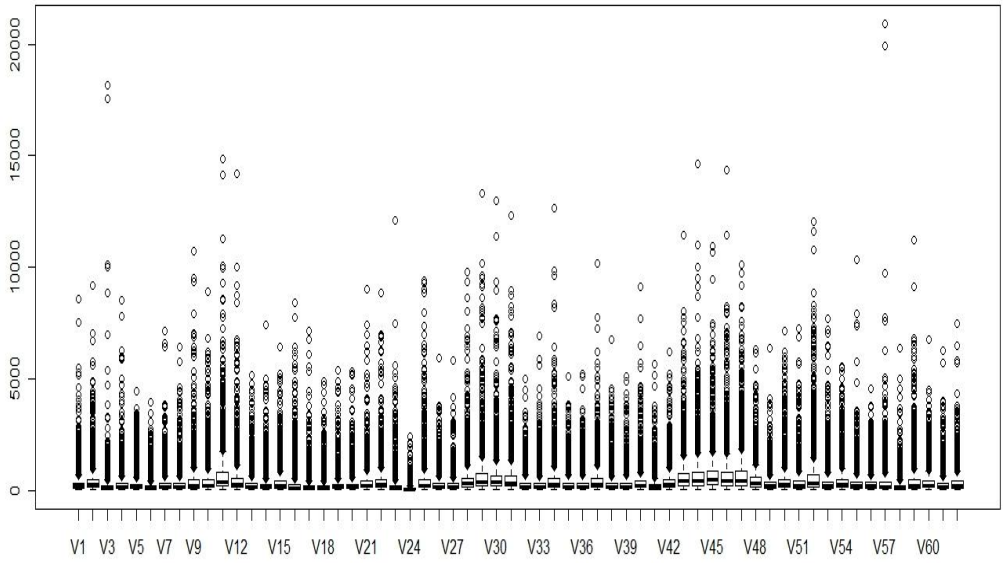

Figure 3.1.1 Boxplot of original data

It is easy to tell for each sample, distribution of genes is not normal, so we take logarithm transformation with base 2 to provide a data which has symmetrical and mostly normal distribution. Interpretation of such a data is more meaningful from biological point of view. Figure 3.1.2 gives boxplot of log-transformed data.

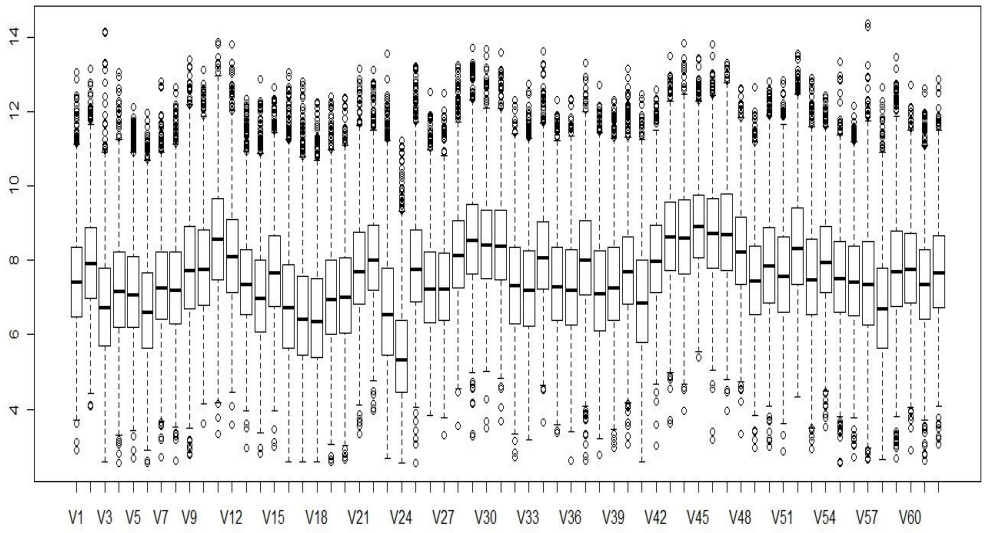

Figure 3.1.2 Boxplot of log-transformed data

The second data preprocessing that we used is standardization, but we used median absolute deviation (MAD) instead of the standard deviation. Boxplot of standardized data is shown in Figure 3.1.3.

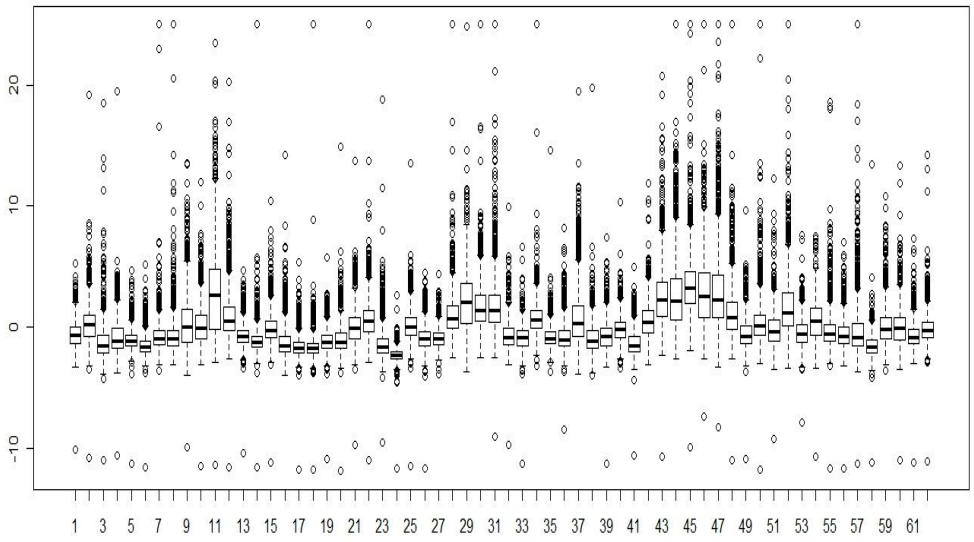

Figure 3.1.3 Boxplot of standardized data

To see the result of standardization clearly, we set the maximum value for y-axis to 25, and boxplot it, but we use the value obtained from standardization to do the clustering experiments later.

Quantile normalization is also used to preprocessing data. In order to obtain better clustering and classification results, we did quantile normalization based on log-transformed data. The result is given by Figure 3.1.4.

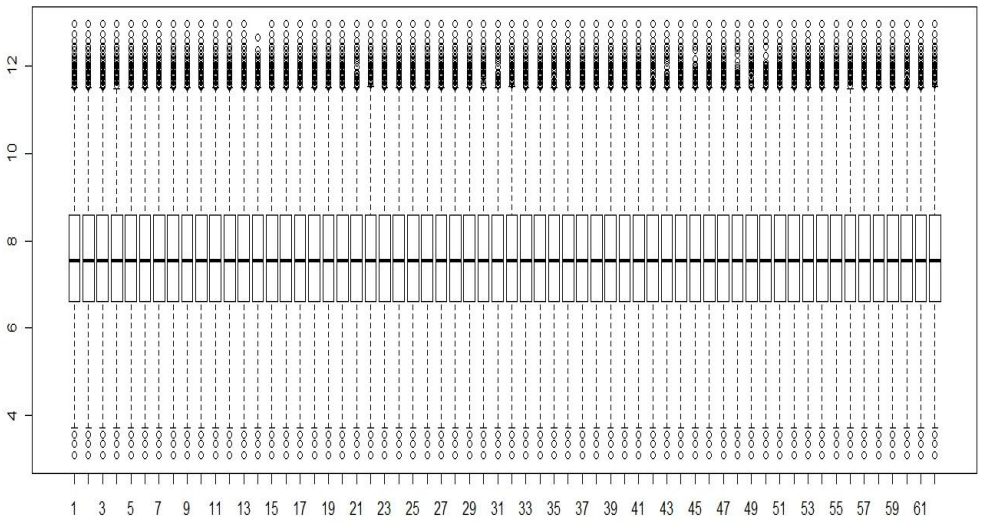

Figure 3.1.4 Boxplot of quantile normalized data

Therefore, we have three preprocessed datasets, log-transformed data, standardized data and quantile normalized data. We applied different clustering and classification methods for the three preprocessed datasets, respectively, and expected to draw some reasonable conclusions.

3.2 Discussion of Clustering Results

DNA microarray is an important tool, which determines the expression of tens of thousands of genes from a sample. However, the data volume it produces can be very large and hard to interpret. Based on the similarity of their expressions, clustering samples can simplify the data, and discover their relationship.

In this research, I utilized K-means, K-medoids with Euclidean and Pearson correlation distance, and Hierarchical clustering with Euclidean and Pearson correlation distance, five ways in total, to group colon tissue samples into two clusters, which are equivalent to tumor samples and normal samples. More specifically, we

chose partitioning around medoidss (PAM) algorithm for K-medoids clustering and bottom-up algorithm for Hierarchical clustering. We repeated five methods that mentioned above on to log-transformation data, quantile normalization data and standardization data, and compared their performance.

Next, I will list results of clustering for three transformed data with size 2000 by 62, size 2000 by 57, and size 2000 by 44 separately, using these five algorithms.

Log-transformed data observation. Table 3.2.1 shows the performance of each algorithm on size 2000 by 62 log-transformed data.

Table 3.1.1 Clustering results of 2000 by 62 log-transformed data

Sample ID	Known Label	2000*62 Log-transformed data				
		K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C
1	1	1	1	1	1	1
2	2	2	2	2	1	1
3	1	1	1	2	2	1
4	2	1	1	2	2	1
5	1	1	1	1	2	2
6	2	1	1	2	2	2
7	1	1	1	1	1	1
8	2	1	1	2	2	2

9	1	2	2	1	1	1
10	2	2	1	2	1	2
11	1	2	2	2	1	2
12	2	2	2	2	1	1
13	1	1	1	1	2	1
14	2	1	1	2	2	1
15	1	2	1	1	1	1
16	2	1	1	1	2	1
17	1	1	1	1	2	2
18	2	1	1	2	2	2
19	1	1	1	1	2	1
20	2	1	1	2	2	2
21	1	2	1	1	1	1
22	2	2	2	2	1	1
23	1	1	1	1	1	2
24	2	1	1	2	2	2
25	1	2	2	1	2	1
26	1	1	1	1	1	1
27	1	1	1	1	1	1
28	1	2	2	1	1	1

29	1	2	2	1	1	1
30	1	2	2	1	1	1
31	1	2	2	1	1	1
32	1	1	1	2	1	2
33	1	1	1	1	1	2
34	1	2	2	1	1	1
35	1	1	1	1	1	1
36	1	1	1	1	2	2
37	1	2	2	2	1	1
38	1	1	1	1	2	2
39	2	1	1	2	1	1
40	1	2	2	1	1	1
41	1	1	1	1	2	1
42	2	2	2	2	1	1
43	2	2	2	2	1	1
44	1	2	2	1	1	2
45	1	2	2	2	1	1
46	1	2	2	1	1	2
47	1	2	2	1	1	2
48	2	2	2	2	1	1

49	1	1	1	2	1	1
50	2	2	1	2	1	1
51	2	1	1	1	1	1
52	1	2	2	1	1	1
53	1	1	1	1	1	1
54	2	2	2	2	1	1
55	2	1	1	2	1	1
56	1	1	1	2	1	1
57	1	1	1	2	2	1
58	1	1	1	1	2	1
59	1	2	1	1	1	1
60	2	2	2	2	1	1
61	1	1	1	1	1	1
62	2	2	1	2	1	1

Samples labeled in orange indicate that it is being grouped into a wrong cluster. How many samples being clustered correctly by each algorithm is shown in Table 3.2.2.

Table 3.2.2 Summary of clustering results of 2000 by 62 log-transformed data

	2000*62 Log-transformed data

	K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C
Ratio between tumor and normal clusters	33:29	39:23	34:28	43:19	45:17
Number of correctly grouped sample	33	33	52	37	35

In the original gene expression data, there are 40 tumor tissue samples and 22 normal tissue samples. According to the table, K-medoids method works relatively better for log-transformed data. Only 10 samples are grouped into wrong cluster, error rate is about 16.13%.

As we know, samples labeled 45,49,51,55 and 56 are contaminated in 62 samples, and they may affect clustering result. Thus, in the following experiments, I removed 5 contaminated samples, and reduced size from 2000 by 62 to 2000 by 57.

The clustering result of 2000 by 57 log-transformed data is shown in Table 3.2.3.

Table 3.2.3 Clustering results of 2000 by 57 log-transformed data

ID	Label	2000*57 Log-transformed data				
		K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C

1	1	1	1	1	1	1
2	2	2	2	1	2	1
3	1	1	1	2	1	1
4	2	1	1	2	1	1
5	1	1	1	2	1	2
6	2	1	1	2	1	2
7	1	1	1	2	1	1
8	2	1	1	2	1	2
9	1	2	2	2	1	1
10	2	2	1	2	2	2
11	1	2	2	2	2	2
12	2	2	2	1	2	1
13	1	1	1	1	1	1
14	2	1	1	2	1	1
15	1	2	2	1	1	1
16	2	1	1	2	1	1
17	1	1	1	2	1	2
18	2	1	1	2	1	2
19	1	1	1	1	1	1
20	2	1	1	2	1	2

21	1	2	2	1	1	1
22	2	2	2	2	2	1
23	1	1	1	2	1	2
24	2	1	1	2	3	2
25	1	2	2	1	1	1
26	1	1	1	1	1	1
27	1	1	1	1	1	1
28	1	2	2	1	2	1
29	1	2	2	1	2	1
30	1	2	2	1	2	1
31	1	2	2	1	2	1
32	1	1	1	2	1	2
33	1	1	1	1	1	2
34	1	2	2	1	2	1
35	1	1	1	1	1	1
36	1	1	1	1	1	2
37	1	2	2	2	2	1
38	1	1	1	1	1	2
39	2	1	1	1	1	1
40	1	2	2	1	1	1

41	1	1	1	1	1	1
42	2	2	2	1	2	1
43	2	2	2	1	2	1
44	1	2	2	1	2	2
46	1	2	2	1	2	2
47	1	2	2	1	2	2
48	2	2	2	2	2	1
50	2	2	2	1	2	1
52	1	2	2	1	2	1
53	1	1	2	1	1	1
54	2	2	2	1	2	1
57	1	1	1	1	1	1
58	1	1	1	1	1	1
59	1	2	2	1	1	1
60	2	2	2	1	2	1
61	1	1	1	1	1	1
62	2	2	2	1	2	1

When Hierarchical method performed to size 2000 by 57 data using

Euclidean method, sample 24 formed a single cluster and isolated from other 56

samples, so we excluded sample 24 in this situation and labeled it as “3”. The experiment is summarized in Table 3.2.4.

Table 3.2.4 Summary of clustering results of 2000 by 57 log-transformed data

	2000*57 Log-transformed data				
	K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C
Ratio between tumor and normal clusters	29:28	29:28	37:20	34:22	40:17
Number of correctly grouped sample	31	29	39	37	32

This set of experiment indicates that K-medoids method using Pearson correlation distance has the best result among five methods. But 13 samples are grouped into wrong cluster. The error rate of 22.81% is higher than that of the K-medoids method using Pearson correlation dissimilarity measurement for size 2000 by 62 log-transformed data.

In original data, 44 samples form 22 pairs. Each pair, one normal and one tumor samples, belongs to one person. I also did experiments using just 22 pairs. Table 3.2.5 gives the clustering result of size 2000 by 44 log-transformed data.

Table 3.2.5 Clustering results of 2000 by 44 log-transformed data

ID	Label	2000*44 Log-transformed data

		K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C
1	1	2	1	1	1	1
2	2	2	2	2	1	1
3	1	1	1	1	2	1
4	2	1	1	2	2	1
5	1	1	1	1	2	2
6	2	1	1	1	2	2
7	1	1	1	1	1	1
8	2	1	1	1	2	2
9	1	2	2	1	1	1
10	2	2	2	2	1	2
11	1	2	2	1	1	2
12	2	2	2	2	1	1
13	1	2	1	1	2	1
14	2	1	1	2	2	1
15	1	2	2	1	1	1
16	2	1	1	1	2	1
17	1	1	1	1	2	2

18	2	1	1	1	2	2
19	1	1	1	1	2	1
20	2	1	1	1	2	2
21	1	2	2	1	1	1
22	2	2	2	2	1	1
23	1	1	1	1	2	2
24	2	1	1	1	2	2
39	2	2	1	2	1	1
40	1	2	2	2	1	1
41	1	1	1	2	2	1
42	2	2	2	2	1	1
43	2	2	2	2	1	1
44	1	2	2	1	1	2
47	1	2	2	1	1	2
48	2	2	2	2	1	1
49	1	2	2	2	1	1
50	2	2	2	2	1	1
51	2	2	2	2	1	1
52	1	2	2	1	1	1
53	1	2	2	2	1	1

54	2	2	2	2	1	1
55	2	2	2	2	1	1
56	1	2	2	2	1	1
59	1	2	2	2	1	1
60	2	2	2	2	1	1
61	1	2	2	2	1	1
62	2	2	2	2	1	1

The ratio between numbers of sample in the two clusters and the numbers of correctly grouped sample are organized in Table 3.2.6.

Table 3.2.6 Summary of clustering results of 2000 by 44 log-transformed data

	2000*44 Log-transformed data				
	K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C
Ratio between tumor and normal clusters	15:29	18:26	21:23	29:15	32:12
Number of correctly grouped sample	21	22	31	23	22

The best result is again achieved by K-medoids method using Pearson correlation distance. 13 samples are grouped into wrong cluster, so the error rate is approximate to 29.55%.

To conclude, for log-transformed gene expression data, K-medoids method using Correlation distance measurement always gives a better clustering result no matter what size the dataset is.

Quantile normalization data observation. I organized the clustering result of five methods for size 2000 by 62, size 2000 by 57, and size 2000 by 44 quantile normalization data in Table 3.2.7.

Table 3.2.7 Clustering results of quantile normalized data

ID	Label	Quantile data 2000*62					Quantile data 2000*57					Quantile data 2000*44				
		K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C	K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C	K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	1	1	1	2	1	1	1	1	2	1	2	1	1
3	1	2	1	1	1	1	2	2	2	1	1	2	1	1	1	1
4	2	2	2	1	1	1	2	2	2	1	1	2	1	2	1	1
5	1	1	1	1	2	2	1	2	2	2	2	1	1	1	2	2
6	2	2	1	1	2	2	2	2	2	2	2	2	1	1	2	2
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	2	2	1	1	2	2	2	2	2	2	2	2	1	1	2	2
9	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1

10	2	2	2	1	2	2	2	2	2	2	2	2	1	2	2	2
11	1	1	1	2	2	2	1	1	2	2	2	1	1	1	2	2
12	2	2	2	1	1	1	2	1	1	1	1	2	2	2	1	1
13	1	1	1	1	1	1	1	2	1	1	1	1	1	2	1	1
14	2	2	2	1	1	1	2	2	2	1	1	2	1	2	1	1
15	1	1	1	1	1	1	2	2	1	1	1	1	1	1	1	1
16	2	1	1	1	1	1	2	2	2	1	1	1	1	1	1	1
17	1	1	1	2	2	2	1	1	2	2	2	1	1	1	2	2
18	2	2	2	2	2	2	2	2	2	2	2	2	1	1	2	2
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20	2	2	1	2	2	2	2	2	2	2	2	2	1	1	2	2
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	2	2	2	1	1	1	2	2	2	1	1	2	1	2	1	1
23	1	1	1	2	2	2	1	2	2	2	2	1	1	1	2	2
24	2	2	2	1	2	2	2	2	2	2	2	2	1	1	2	2
25	1	1	1	1	2	2	1	1	1	2	2					
26	1	1	1	1	1	1	1	1	1	1	1					
27	1	1	2	1	1	1	1	1	1	1	1					
28	1	1	1	1	1	1	1	1	1	1	1					
29	1	1	1	1	1	1	1	1	1	1	1					

30	1	1	1	1	1	1	1	1	1	1	1					
31	1	1	1	1	1	1	1	1	1	1	1					
32	1	1	1	2	2	2	1	1	2	2	2					
33	1	1	1	2	2	2	1	1	1	2	2					
34	1	1	1	1	1	1	1	1	1	1	1					
35	1	1	1	1	1	1	1	1	1	1	1					
36	1	1	1	2	2	2	1	1	1	2	2					
37	1	1	1	1	1	1	1	1	2	1	1					
38	1	1	1	2	2	2	1	1	1	2	2					
39	2	2	2	1	1	1	2	2	1	1	1	2	2	2	1	1
40	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1
41	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1
42	2	2	2	2	1	1	2	2	1	1	1	2	2	2	1	1
43	2	2	2	2	1	1	2	1	1	1	1	2	2	2	1	1
44	1	1	1	2	2	2	1	1	1	2	2	1	2	1	2	2
45	1	2	2	1	1	1										
46	1	1	1	2	2	2	1	1	1	2	2					
47	1	1	1	2	2	2	1	1	1	2	2	1	1	1	2	2
48	2	2	2	1	1	1	2	1	2	1	1	2	2	2	1	1
49	1	2	2	1	1	1						2	2	2	1	1

50	2	2	2	1	1	1	2	2	1	1	1	2	2	2	1	1
51	2	1	1	1	1	1						1	1	2	1	1
52	1	1	1	2	1	1	1	1	1	1	1	1	2	1	1	1
53	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1
54	2	2	2	1	1	1	2	2	1	1	1	2	2	2	1	1
55	2	1	1	1	1	1						1	2	2	1	1
56	1	2	2	2	1	1						2	2	2	1	1
57	1	2	1	1	1	1	2	2	2	1	1					
58	1	1	1	1	1	1	1	1	1	1	1					
59	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1
60	2	2	2	1	1	1	2	1	1	1	1	2	2	2	1	1
61	1	1	1	2	1	1	1	1	1	1	1	1	2	2	1	1
62	2	2	2	1	1	1	2	1	1	1	1	2	2	2	1	1

I also summarized the ratios between the two cluster sizes and the numbers of correctly grouped sample in Table 3.2.8.

Table 3.2.8 Summary of clustering results of quantile normalized data

		Ratio between tumor and normal clusters	Number of correctly grouped sample
Quantile Data 2000*62	K-Means	38:24	54
	K-Medoids.E	42:20	52
	K-Medoids.C	45:17	31
	Hclust.E	44:18	34
	Hclust.C	44:18	34
Quantile Data 2000*57	K-Means	34:23	54
	K-Medoids.E	37:20	45

	K-Medoids.C	37:20	39
	Hclust.E	39:18	31
	Hclust.C	39:18	31
Quantile Data 2000*44	K-Means	22:22	38
	K-Medoids.E	26:18	24
	K-Medoids.C	20:24	30
	Hclust.E	32:12	22
	Hclust.C	32:12	22

Performing clustering on quantile normalized data, results of Hierarchical method are exactly the same no matter is Euclidean dissimilarity or Pearson correlation dissimilarity used. This could be proved by the following dendrograms Figure 3.2.1, Figure 3.2.2, and Figure 3.2.3.

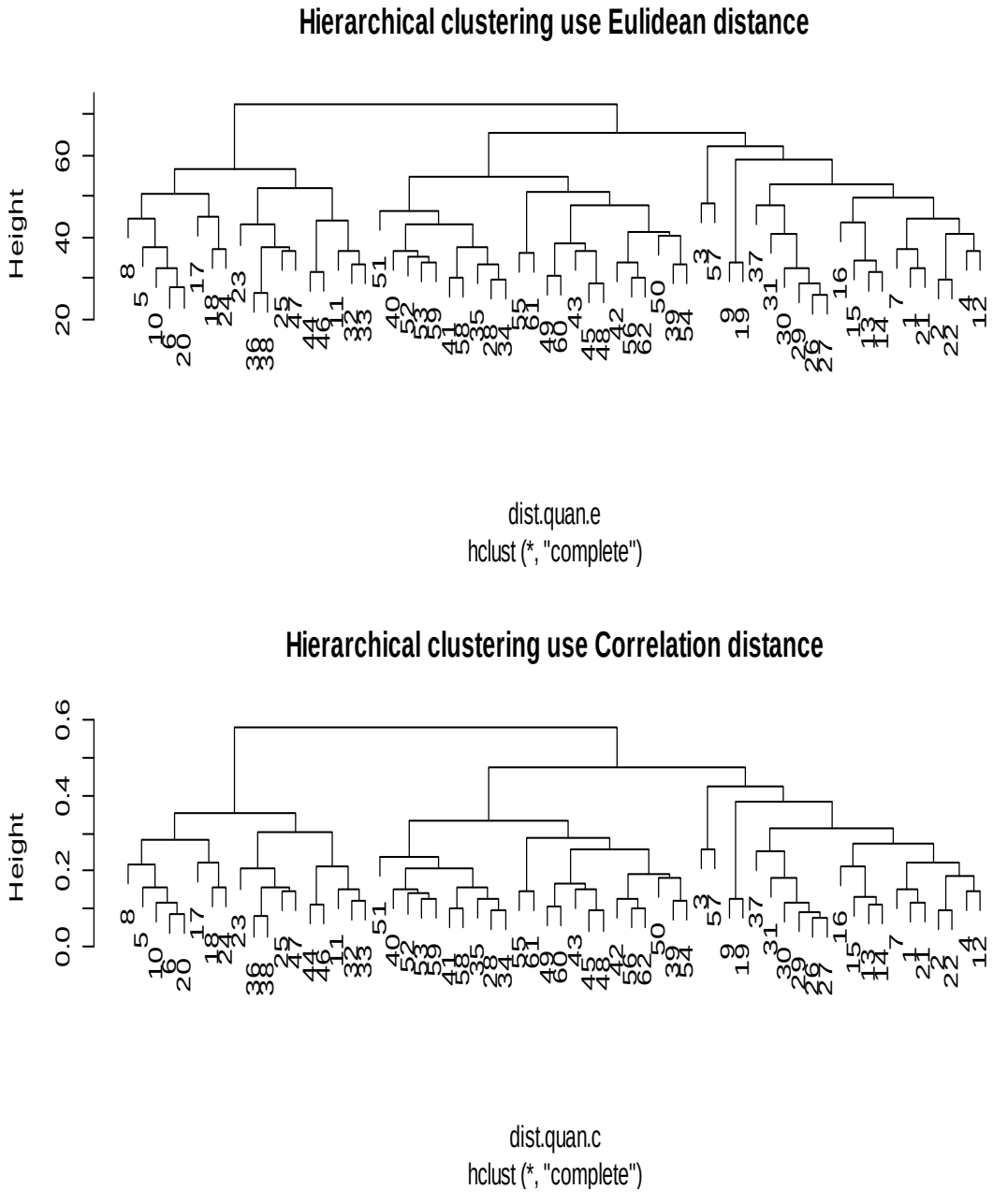

Figure 3.2.1 Hierarchical clustering with Euclidean and correlation distance of 2000 by 62 quantile normalized data

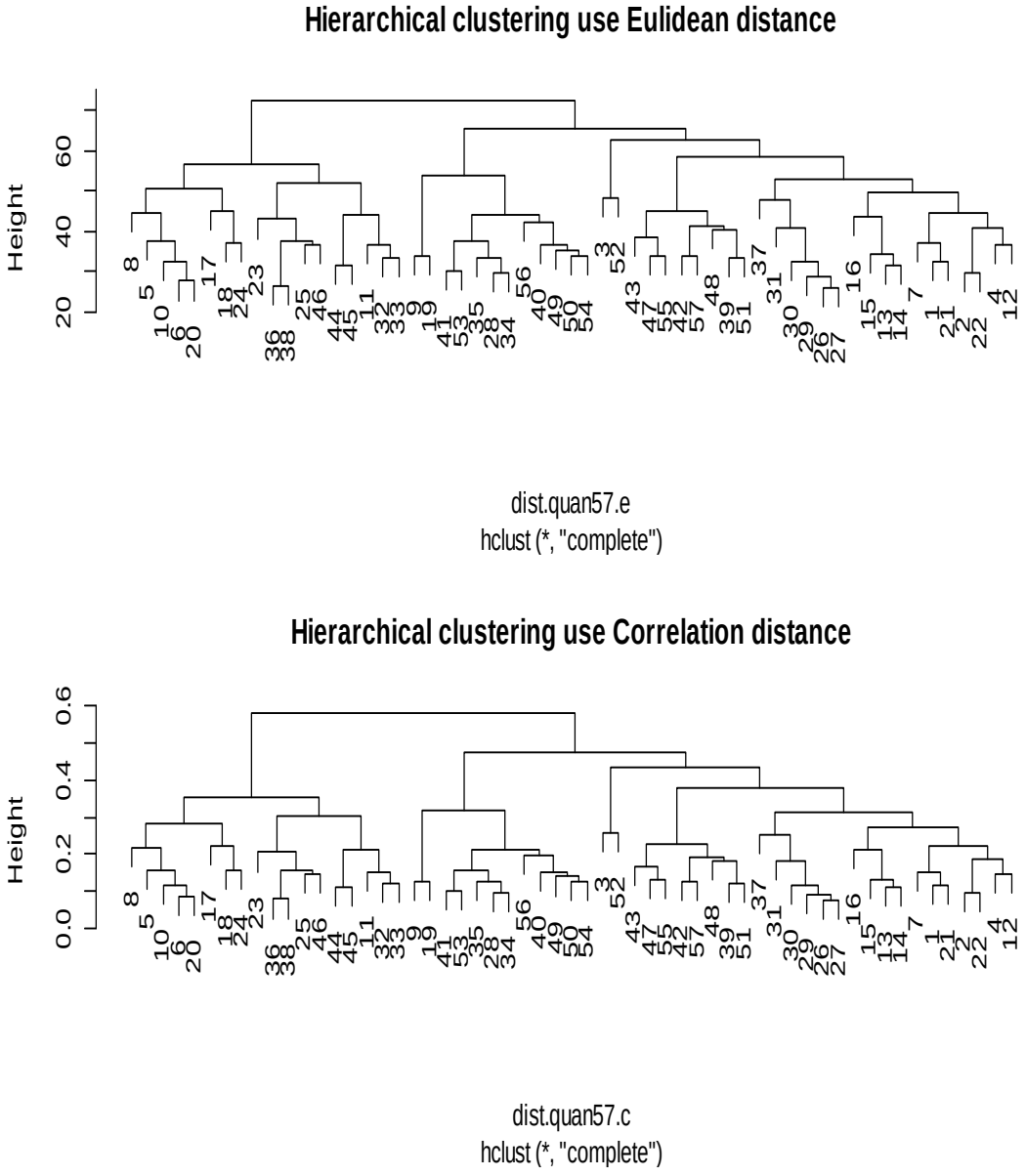

Figure 3.2.2 Hierarchical clustering with Euclidean and correlation distance of 2000 by 57 quantile normalized data

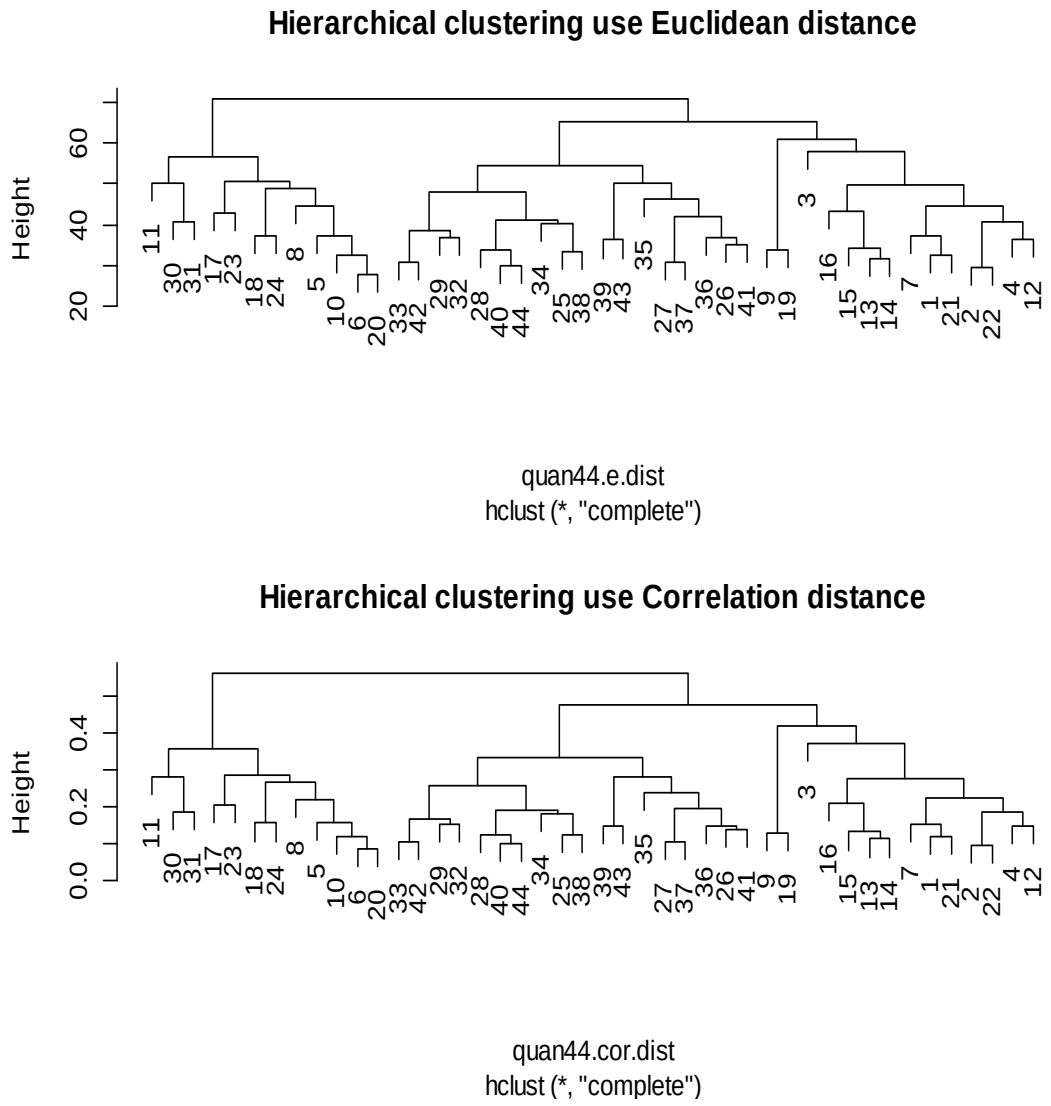

Figure 3.2.3 Hierarchical clustering with Euclidean and correlation distance of 2000 by 44 quantile normalized data

In addition, no matter what size of quantile normalized data is, K-means method always obtains the best clustering among these five different algorithms. Their error rates are 12.90%, 5.26% and 13.64%, respectively. Especially, for size 2000 by 57 quantile normalized data, which removed 5 contaminative samples, only 3 samples are grouped into wrong cluster. It is also the best clustering result in all the

experiments performed. Although, for size 2000 by 62 quantile normalized data, 8 samples are grouped into wrong cluster, 5 of them are contaminative samples. This implies clustering using 2000 by 62 quantile normalized data does not affect by contaminative samples seriously in this case.

Standardization using MAD data observation. I organized clustering results for standardization data of 3 different sizes using MAD instead of standard deviation in Table 3.2.9.

Table 3.2.9 Clustering results of standardized data

ID	Label	Standardization 2000*62					Standardization 2000*57					Standardization 2000*44				
		K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C	K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C	K-Means	K-Medoids.E	K-Medoids.C	Hclust.E	Hclust.C
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2
3	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
4	2	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
5	1	1	1	2	1	2	1	1	2	1	2	1	1	1	1	2
6	2	1	1	2	1	2	1	1	2	1	2	1	1	1	1	2
7	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1
8	2	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1

9	1	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1
10	2	1	1	2	1	2	1	1	2	1	2	1	1	1	1	2
11	1	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2
12	2	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2
13	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
14	2	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
15	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
16	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
18	2	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
19	1	1	1	2	1	2	1	1	2	1	2	1	1	1	1	1
20	2	1	1	2	1	2	1	1	2	1	2	1	1	1	1	2
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	2	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
23	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
24	2	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
25	1	1	1	2	1	2	1	1	2	1	2					
26	1	1	1	1	1	1	1	1	1	1	1					
27	1	1	1	1	1	1	1	1	1	1	1					
28	1	1	1	1	1	2	2	1	1	1	1					

29	1	2	1	1	2	1	2	1	1	2	1					
30	1	2	1	1	2	2	2	1	1	2	1					
31	1	2	1	1	2	1	2	1	1	2	1					
32	1	1	1	2	1	2	1	1	2	1	2					
33	1	1	1	2	1	2	1	1	2	1	2					
34	1	1	1	1	1	2	2	1	1	1	1					
35	1	1	1	1	1	2	1	1	1	1	1					
36	1	1	1	2	1	2	1	1	2	1	2					
37	1	1	1	1	2	1	1	1	1	2	1					
38	1	1	1	1	1	2	1	1	1	1	2					
39	2	1	1	2	1	1	1	1	2	1	1	1	1	2	1	1
40	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
41	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
42	2	1	1	2	1	1	1	1	2	1	1	1	1	2	1	2
43	2	2	2	2	2	1	2	2	2	2	1	2	2	2	2	2
44	1	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1
45	1	2	2	2	2	1										
46	1	2	2	2	2	2	2	2	2	2	2					
47	1	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1
48	2	2	1	2	2	1	2	1	2	2	1	1	1	2	2	2

49	1	1	1	2	1	1						1	1	2	1	2
50	2	1	1	2	1	1	1	1	2	1	1	1	1	2	1	2
51	2	1	1	2	1	1						1	1	1	1	1
52	1	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1
53	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
54	2	1	1	2	1	1	1	1	2	1	1	1	1	2	1	1
55	2	1	1	2	1	1						1	1	2	1	1
56	1	1	1	2	1	1						1	1	2	1	2
57	1	1	1	2	1	1	1	1	2	1	1					
58	1	1	1	1	1	1	1	1	1	1	1					
59	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
60	2	1	1	2	1	1	1	1	2	1	1	1	1	2	1	2
61	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1
62	2	1	1	2	1	1	1	1	2	1	1	1	1	2	1	2

Ratio between sizes of the two clusters and numbers of sample is grouped correctly are summarized in Table 3.2.10.

Table 3.2.10 Summary of clustering results of standardized data

		Ratio between tumor and normal clusters	Number of correctly grouped sample
Standardization 2000*62	K-Means	51:11	33
	K-Medoids.E	55:7	35
	K-Medoids.C	21:41	39
	Hclust.E	50:12	32
	Hclust.C	40:22	26

Standardization 2000*57	K-Means	45:12	29
	K-Medoids.E	51:6	33
	K-Medoids.C	21:36	37
	Hclust.E	46:11	30
	Hclust.C	41:16	27
Standardization 2000*44	K-Means	39:5	19
	K-Medoids.E	39:5	19
	K-Medoids.C	33:11	29
	Hclust.E	38:6	20
	Hclust.C	29:15	29

This set of experiments of standardization data using MAD gave a worse clustering result than those of log-transformed data and quantile normalized data. But comparatively, K-mediods method using Correlation dissimilarity always gets a better clustering result than the other 4 algorithms. The error rates are about 37.10%, 35.09%, and 34.09%, respectively.

In conclusion, for logarithm transformed data and standardized data which using MAD, K-medoids method using Pearson correlation as dissimilarity measurement gives relative better clustering results among all 5 clustering algorithms. Furthermore, the best clustering result is obtained when performing K-means algorithm on quantile normalization data. Particularly, after I removed 5 contaminative samples, this algorithm successfully clusters 54 samples out of 57 samples, an accuracy rate of 94.74%.

3.3 Comparison of Classification Results

When one patient was suspected of having cancer, the doctor always tests blood or tissue samples that are extracted from the patient in order to determine

whether the patient has cancer or not. I also want to predict the result using mathematical methods. In this research, two different classification algorithms are adopted. They are linear discriminant analysis, and minimum sum of squared-error.

In this research, I separate the 62 samples into training and testing two sets. Training set is used to train the predictor, and testing set is used to examine how good the predictor is.

At the very beginning, I used 40 samples in the training set, and 22 samples in the testing set. I performed the two classification methods on the three transformed datasets, and summarized forecasting results in Table 3.3.1as following.

Table 3.3.1 Classification results of training set 42 vs. testing set 20

ID	Label	Linear discriminant analysis			Minimum SSE		
		logarithm	Quantile	standardize	logarithm	quantile	standardize
41	1	1	1	1	1	1	1
42	2	2	2	2	2	2	2
43	2	2	2	2	2	2	2
44	1	1	1	1	1	1	1
45	1	2	2	2	2	2	2
46	1	1	1	1	1	1	1

47	1	1	1	1	1	1	1
48	2	2	2	2	2	2	2
49	1	2	2	2	2	2	2
50	2	2	2	2	2	2	2
51	2	1	1	1	1	1	1
52	1	1	1	1	1	1	1
53	1	1	1	1	1	1	1
54	2	2	2	2	2	2	2
55	2	1	1	1	1	1	1
56	1	2	2	2	2	2	2
57	1	1	1	1	1	1	1
58	1	1	1	1	1	1	1
59	1	1	1	1	1	1	1
60	2	2	2	2	2	2	2
61	1	1	1	1	1	1	1
62	2	2	2	2	2	2	2

5 out of the 20 samples were predicted incorrectly in each experiment for linear discriminant analysis and minimum sum of squared-error method. Notice that, those 5 samples, which grouped by mistake, are all contaminative samples. Therefore,

from this point of view, both linear discriminant analysis and minimum sum of squared-error methods predicted the classes accurately.

However, in this way, I can neither see which method is better, not which transformation of the data kept more information of original data. Thus, I reduced the size of training set down to 20 samples, and increased the size of testing samples to 42. Table 3.3.2 sums up the classification results for each experiment.

Table 3.3.2 Classification results of training set 20 vs. testing set 42

ID	Label	Linear discriminant analysis			Minimum SSE		
		Logarithm	Quantile	standardize	logarithm	quantile	standardize
21	1	1	1	1	1	1	1
22	2	2	2	2	2	2	2
23	1	1	1	1	1	1	1
24	2	2	2	1	2	2	2
25	1	1	1	1	1	1	1
26	1	1	1	1	1	1	1
27	1	1	1	1	1	1	2
28	1	1	1	1	1	1	1
29	1	1	1	1	1	1	1

30	1	1	1	1	1	1	2
31	1	1	1	1	1	1	1
32	1	1	1	1	1	1	1
33	1	1	1	1	1	1	1
34	1	1	1	1	1	1	1
35	1	1	1	1	1	1	1
36	1	1	1	1	1	1	1
37	1	1	1	1	1	1	1
38	1	1	1	1	1	1	1
39	2	2	2	2	2	2	2
40	1	1	1	1	1	1	1
41	1	1	1	1	1	1	1
42	2	2	2	1	2	2	2
43	2	2	2	2	2	2	2
44	1	1	1	1	1	1	1
45	1	2	2	2	2	2	2
46	1	1	1	1	1	1	1
47	1	1	1	1	1	1	1
48	2	2	2	2	2	2	2
49	1	2	2	2	2	2	2

50	2	2	2	2	2	2	2
51	2	1	1	1	1	1	1
52	1	1	1	1	1	1	1
53	1	2	1	1	1	1	1
54	2	2	2	2	2	2	2
55	2	1	1	1	1	1	1
56	1	2	2	2	2	2	2
57	1	2	2	1	1	1	1
58	1	1	1	1	1	1	1
59	1	1	1	1	1	1	1
60	2	2	2	2	2	2	2
61	1	1	1	1	1	1	1
62	2	2	2	2	2	2	2

Predictors trained by reduced training set show different classification outputs. Let's summary how many samples are classified incorrectly in this set of experiments in Table 3.3.3.

Table 3.3.3 Summary of classification results of training set 20 vs. testing set 42

Linear discriminant analysis	Logarithm transformation	7
	Quantile normalization	6
	standardization	7
Minimum sum of squared-error	Logarithm transformation	5
	Quantile normalization	5
	standardization	7

After such a contrast, minimum sum of squared-error algorithm obtains better or the same classification results than the linear discriminant analysis. Moreover, by contrast, quantile normalization is a better transformation method. Although, quantile normalized data and log-transformed data give exactly the same prediction by using minimum sum of squared-error, quantile normalized data provide a better forecast when linear discriminant analysis is applied.

Chapter 4 Conclusions

In this research, some conclusions and laws could be drawn from the experimental results.

1. In general, among three data preprocessing methods, quantile normalization method gives the better clustering result than the other two methods, regardless of which clustering algorithm has been applied on which size of data.

2. For quantile normalized data, K-means algorithm always obtains the best clustering result no matter the size of transformed data. In addition, the best clustering result that I have gotten in all experiments is from K-means algorithm on size 2000 by 57 quantile normalized data with a 94.74% accuracy.

3. For log-transformed data and standardized data, K-mediod algorithm based on Pearson correlation distance metric has the best clustering performance among the 5 clustering algorithms. Compared to standardization method, log-transformation is a relatively better data preprocessing method.

4. When I separated 62 samples into 40 training samples and 22 testing samples, both of linear discriminant analysis and minimum sum of squared-error algorithm provide consistent and good results. The predictor trained by these two method correctly classified all testing samples except the 5 contaminative ones.

5. In order to compare the methods, 20 samples was left in training set, and testing set was expanded to 42 samples. Obviously, minimum sum of squared-error algorithm is superior to linear discriminant analysis. The result also confirmed that quantile normalization is a better data preprocessing method for colon cancer data analysis.

As the tumor microarray data have so many features but usually come with limited number of samples and noise, it has all kind of challenges in analyzing the data.

Bibliography

[A.U.] Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., & Levine, A. J. (1999). *Data pertaining to the article 'Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays'*. Retrieved from http://genomics-pubs.princeton.edu/oncology/affydata/index.html

[D.O.] Duda, R. O., Hart, P. E., & Stork, D. G. (2001) *Pattern classification*. New York, NY: John Wiley & Sons, Inc.

[D.S.] Drăghici, S. (2003). *Data analysis tools for DNA microarrays.* New York, NY: Chapman & Hall/CRC.

[K.B.] Korol, A. B. (2003). *Microarray cluster analysis and applications*. Retrieved from http://www.science.co.il/enuka/Essays/Microarray-Review.pdf

[S.] Scholarpedia. (2009). *K-nearest_neighbor.* Retrieved from http://www.scholarpedia.org/article/K-nearest_neighbor

[W.M.] Wolfram Mathematica. (2011). Neural Networks Document*ation: 2.2 Data Preprocessing*. Retrieved from http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/2.2.0.html

[X.X.] Xu, W. X., You, C. X., Xu, J. W., & Chen, Y. R. (2010). *Biochip Platforms for DNA Diagnostics* [PowerPoint slides]. Retrieved from http://www.slidefinder.net/B/Biochip_Platforms_DNA_Diagnostics_%E8%A8%B1%E6%96%87%E9%A6%A8/10935617

Appendix: R code

Program One: Logarithm transformation and clustering analysis.

```
rm(list=ls(all=TRUE)) # clear all previous data

library(cluster)

library(MASS)

# read in the data

colon.dat<-read.table("C:/Users/bo.liu/Desktop/R/I2000.txt",header=FALSE,sep=' ')

colon.dat<-data.matrix(colon.dat,main='Boxplot of Original Data')

boxplot(colon.dat)

# 1.pre-processing:log-transformation data

logcolon.dat<-log2(colon.dat)

logcolon.dat<-data.matrix(logcolon.dat)

boxplot(logcolon.dat,main='Boxplot of Log-transformed Data')

# 2.1.kmeans clustering(2000 by 62)

tlog<-t(logcolon.dat)

set.seed(17)

km.log<-kmeans(tlog,2)

km.log
```

```
# 2.2.kmediod clustering-Partitioning Around Medoids(2000 by 62)

pam.log.e<-pam(tlog,2,diss=F,metric="euclidean")

summary(pam.log.e)

log.corr<-cor(logcolon.dat,method = "pearson")

dist.log.c<-matrix(nrow=62,ncol=62)

        for(i in 1:62){

        dist.log.c[i,]<-c(1:62)

        for(j in 1:62){

        dist.log.c[i,j]<-1-log.corr[i,j]}}

dist.log.c<-as.dist(dist.log.c,diag=F,)

pam.log.corr<-pam(dist.log.c,2,diss=F,cluster.only=F,do.swap=F,keep.diss=F,keep.da

        ta=F)

summary (pam.log.corr)

# 2.3.Hierarchical Cluster Analysis - Euclidean (2000 by 62)

dist.log.e<-dist(tlog,method="euclidean")

hclust.log.e<-hclust(dist.log.e,method='complete')

plot(hclust.log.e,labels=NULL)

##      Hierarchical Cluster Analysis - Correlation (logcolon.dat)

hclust.log.corr<-hclust(dist.log.c,method='complete')

plot(hclust.log.corr,labels=NULL)
```

```
# 3.reduce the matrix to 2000*57 (delete contaminate ones)

log57.dat<-logcolon.dat[,-c(45,49,51,55,56)]

# 4.1.kmeans clustering(2000 by 57)

tlog57<-t(log57.dat)

set.seed(23)

km.log57<-kmeans(tlog57,2)

km.log57

# 4.2.kmediod clustering-Partitioning Around Medoids(2000 by 57)

pam.log57.e<-pam(tlog57,2,diss=F,metric="euclidean")

summary(pam.log57.e)

log57.corr<-cor(log57.dat,method = "pearson")

dist.log57.c<-matrix(nrow=57,ncol=57)

        for(i in 1:57){

        dist.log57.c[i,]<-c(1:57)

        for(j in 1:57){

        dist.log57.c[i,j]<-1-log57.corr[i,j]}}

dist.log57.c<-as.dist(dist.log57.c,diag=F,)

pam.log57.corr<-pam(dist.log57.c,2,diss=F,cluster.only=F,do.swap=F,keep.diss=F,ke

        ep.data=F)

summary (pam.log57.corr)
```

```
# 4.3.Hierarchical Cluster Analysis - Euclidean (2000 by 57)
dist.log57.e<-dist(tlog57,method="euclidean")
hclust.log57.e<-hclust(dist.log57.e,method='complete')
plot(hclust.log57.e,labels=NULL)
##      Hierarchical Cluster Analysis - Correlation (logcolon.dat)
hclust.log57.corr<-hclust(dist.log57.c,method='complete')
plot(hclust.log57.corr,labels=NULL)
# 5.reduce the matrix to 57*44 (22 pairs)
log44.dat<-logcolon.dat[,-c(25:38,45,46,57,58)]
boxplot(log44.dat)
# 6.1.kmeans clustering (2000*44)
tlog44<-t(log44.dat)
set.seed(104)
kmeans.log44<-kmeans(tlog44,2,iter.max=5)
kmeans.log44
# 6.2.kmediod clustering-Partitioning Around Medoids(2000*44)
pam.log44.e<-pam(tlog44,2,diss=F,metric="euclidean")
summary(pam.log44.e)
log44.cor<-cor(log44.dat,method = "pearson")
log44.cor.dist<-matrix(nrow=44,ncol=44)
```

```
        for(i in 1:44){
        log44.cor.dist[i,]<-c(1:44)
        for(j in 1:44){
        log44.cor.dist[i,j]<- 1-log44.cor[i,j]}}
log44.cor.dist<-as.dist(log44.cor.dist,diag=F,)
pam.log44.cor<-pam(log44.cor.dist,2,diss=F,cluster.only=F,do.swap=F,keep.diss=F,k
        eep.data=F)
summary (pam.log44.cor)
# 6.3.Hierarchical Cluster Analysis - Euclidean (2000*44)
log44.e.dist<-dist(tlog44,method="euclidean")
hclust.log44.e<-hclust(log44.e.dist,method='complete')
plot(hclust.log44.e,labels=NULL)
##      Hierarchical Cluster Analysis - Correlation (2000*44)
hclust.log44.cor<-hclust(log44.cor.dist,method='complete')
plot(hclust.log44.cor,labels=NULL)
```

Program Two: Quantile normalization and clustering analysis.

```
rm(list=ls(all=TRUE)) # clear all previous data
library(cluster)
library(MASS)
# read in the data
```

```
colon.dat<-read.table("C:/Users/bo.liu/Desktop/R/I2000.txt",header=FALSE,sep=' ')
colon.dat<-data.matrix(colon.dat)
boxplot(colon.dat,main='Boxplot of Original Data')
# 1.pre-processing:Quantile normalization data
library(affy)
library(preprocessCore)
logcolon.dat<-log2(colon.dat)
quantile.dat<-normalize.quantiles(logcolon.dat,copy=TRUE)
boxplot(quantile.dat,main='Boxplot of Quantile Normalized Data')
# 2.1.kmeans clustering(2000 by 62)
tquan<-t(quantile.dat)
set.seed(18)
km.quan<-kmeans(tquan,2,iter.max=10)
km.quan
# 2.2.kmediod clustering-Partitioning Around Medoids(2000 by 62)
pam.quan.e<-pam(tquan,2,diss=F,metric="euclidean")
summary(pam.quan.e)
quan.corr<-cor(quantile.dat,method = "pearson")
dist.quan.c<-matrix(nrow=62,ncol=62)
        for(i in 1:62){
```

```
        dist.quan.c[i,]<-c(1:62)
        for(j in 1:62){
        dist.quan.c[i,j]<-1-quan.corr[i,j]}}
dist.quan.c<-as.dist(dist.quan.c,diag=F,)
pam.quan.corr<-pam(dist.quan.c,2,diss=F,cluster.only=F,do.swap=F,keep.diss=F,kee
        p.data=F)
summary (pam.quan.corr)
# 2.3.Hierarchical Cluster Analysis - Euclidean (2000 by 62)
dist.quan.e<-dist(tquan,method="euclidean")
hclust.quan.e<-hclust(dist.quan.e,method='complete')
plot(hclust.quan.e,labels=NULL,main='Hierarchical clustering use Eulidean distance')
##      Hierarchical Cluster Analysis - Correlation (2000 by 62)
hclust.quan.corr<-hclust(dist.quan.c,method='complete')
plot(hclust.quan.corr,labels=NULL,main='Hierarchical clustering use Correlation
        distance')
# 3.reduce the matrix to 2000*57 (delete contaminate ones)
quan57.dat<-quantile.dat[,-c(45,49,51,55,56)]
# 4.1.kmeans clustering(2000 by 57)
tquan57<-t(quan57.dat)
set.seed(8)
```

```
km.quan57<-kmeans(tquan57,2)

km.quan57

# 4.2.kmediod clustering-Partitioning Around Medoids(2000 by 57)

pam.quan57.e<-pam(tquan57,2,diss=F,metric="euclidean")

summary(pam.quan57.e)

quan57.corr<-cor(quan57.dat,method = "pearson")

dist.quan57.c<-matrix(nrow=57,ncol=57)

        for(i in 1:57){

        dist.quan57.c[i,]<-c(1:57)

        for(j in 1:57){

        dist.quan57.c[i,j]<-1-quan57.corr[i,j]}}

dist.quan57.c<-as.dist(dist.quan57.c,diag=F,)

pam.quan57.corr<-pam(dist.quan57.c,2,diss=F,cluster.only=F,do.swap=F,keep.diss=F

        ,keep.data=F)

summary (pam.quan57.corr)

# 4.3.Hierarchical Cluster Analysis - Euclidean (2000 by 57)

dist.quan57.e<-dist(tquan57,method="euclidean")

hclust.quan57.e<-hclust(dist.quan57.e,method='complete')

plot(hclust.quan57.e,labels=NULL,main='Hierarchical clustering use Eulidean

        distance')
```

```
##      Hierarchical Cluster Analysis - Correlation (2000 by 57)
hclust.quan57.corr<-hclust(dist.quan57.c,method='complete')
plot(hclust.quan57.corr,labels=NULL,main='Hierarchical clustering use Correlation
            distance')
# 5.reduce the matrix to 57*44 (22 pairs)
quan44.dat<-quantile.dat[,-c(25:38,45,46,57,58)]
boxplot(quan44.dat)
# 6.1.kmeans clustering (2000*44)
tquan44<-t(quan44.dat)
set.seed(10)
kmeans.quan44<-kmeans(tquan44,2,iter.max=5)
kmeans.quan44
# 6.2.kmediod clustering-Partitioning Around Medoids(2000*44)
pam.quan44.e<-pam(tquan44,2,diss=F,metric="euclidean")
summary(pam.quan44.e)
quan44.cor<-cor(quan44.dat,method = "pearson")
quan44.cor.dist<-matrix(nrow=44,ncol=44)
            for(i in 1:44){
            quan44.cor.dist[i,]<-c(1:44)
            for(j in 1:44){
```

```
        quan44.cor.dist[i,j]<- 1-quan44.cor[i,j]}}
quan44.cor.dist<-as.dist(quan44.cor.dist,diag=F,)
pam.quan44.cor<-pam(quan44.cor.dist,2,diss=F,cluster.only=F,do.swap=F,keep.diss=
        F,keep.data=F)
summary (pam.quan44.cor)
# 6.3.Hierarchical Cluster Analysis - Euclidean (2000*44)
quan44.e.dist<-dist(tquan44,method="euclidean")
hclust.quan44.e<-hclust(quan44.e.dist,method='complete')
plot(hclust.quan44.e,labels=NULL,main='Hierarchical clustering use Euclidean
        distance')
##      Hierarchical Cluster Analysis - Correlation (2000*44)
hclust.quan44.cor<-hclust(quan44.cor.dist,method='complete')
plot(hclust.quan44.cor,labels=NULL,main='Hierarchical clustering use Correlation
        distance')
```

Program Three: Standardization data preprocessing and clustering analysis.

```
rm(list=ls(all=TRUE)) # clear all previous data
library(cluster)
library(MASS)
# read in the data
colon.dat<-read.table("C:/Users/bo.liu/Desktop/R/I2000.txt",header=FALSE,sep=' ')
```

```
colon.dat<-data.matrix(colon.dat,main='Boxplot of Original Data')
boxplot(colon.dat)
# 1.pre-processing:MAD data
mad.dat<-c(1:2000)
        for (i in 1:2000){
        mad.dat[i]<-mad(colon.dat[i,], constant=1)}
mean.dat<-c(1:2000)
        for(i in 1:2000){
        mean.dat[i]<-mean(colon.dat[i,])}
stand.dat<-matrix(nrow=2000,ncol=62)
        for(i in 1:2000){
        stand.dat[i,]<-c(1:62)
        for (j in 1:62){
        stand.dat[i,j]<-(colon.dat[i,j]-mean.dat[i])/mad.dat[i]}}
stand.dat<-data.matrix(stand.dat)
range(stand.dat)
which(stand.dat>25)
#stand.dat[stand.dat>25]<-25
boxplot(stand.dat,horizontal=F,main='Boxplot of Standardized Data')
# 2.1.kmeans clustering(2000 by 62)
```

```
tstand<-t(stand.dat)

set.seed(18)

km.stand<-kmeans(tstand,2,iter.max=10)

km.stand

# 2.2.kmediod clustering-Partitioning Around Medoids(2000 by 62)

pam.stand.e<-pam(tstand,2,diss=F,metric="euclidean")

summary(pam.stand.e)

stand.corr<-cor(stand.dat,method = "pearson")

dist.stand.c<-matrix(nrow=62,ncol=62)

        for(i in 1:62){

        dist.stand.c[i,]<-c(1:62)

        for(j in 1:62){

        dist.stand.c[i,j]<-1-stand.corr[i,j]}}

dist.stand.c<-as.dist(dist.stand.c,diag=F,)

pam.stand.corr<-pam(dist.stand.c,2,diss=F,cluster.only=F,do.swap=F,keep.diss=F,kee

        p.data=F)

summary (pam.stand.corr)

# 2.3.Hierarchical Cluster Analysis - Euclidean (2000 by 62)

dist.stand.e<-dist(tstand,method="euclidean")

hclust.stand.e<-hclust(dist.stand.e,method='complete')
```

```
plot(hclust.stand.e,labels=NULL)

##      Hierarchical Cluster Analysis - Correlation (2000 by 62)

hclust.stand.corr<-hclust(dist.stand.c,method='complete')

plot(hclust.stand.corr,labels=NULL)

# 3.reduce the matrix to 2000*57 (delete contaminate ones)

stand57.dat<-stand.dat[,-c(45,49,51,55,56)]

# 4.1.kmeans clustering(2000 by 57)

tstand57<-t(stand57.dat)

set.seed(22)

km.stand57<-kmeans(tstand57,2)

km.stand57

# 4.2.kmediod clustering-Partitioning Around Medoids(2000 by 57)

pam.stand57.e<-pam(tstand57,2,diss=F,metric="euclidean")

summary(pam.stand57.e)

stand57.corr<-cor(stand57.dat,method = "pearson")

dist.stand57.c<-matrix(nrow=57,ncol=57)

        for(i in 1:57){

        dist.stand57.c[i,]<-c(1:57)

        for(j in 1:57){

        dist.stand57.c[i,j]<-1-stand57.corr[i,j]}}
```

```
dist.stand57.c<-as.dist(dist.stand57.c,diag=F,)
pam.stand57.corr<-pam(dist.stand57.c,2,diss=F,cluster.only=F,do.swap=F,keep.diss=
        F,keep.data=F)
summary (pam.stand57.corr)
# 4.3.Hierarchical Cluster Analysis - Euclidean (2000 by 57)
dist.stand57.e<-dist(tstand57,method="euclidean")
hclust.stand57.e<-hclust(dist.stand57.e,method='complete')
plot(hclust.stand57.e,labels=NULL)
##      Hierarchical Cluster Analysis - Correlation (2000 by 57)
hclust.stand57.corr<-hclust(dist.stand57.c,method='complete')
plot(hclust.stand57.corr,labels=NULL)
# 5.reduce the matrix to 57*44 (22 pairs)
stand44.dat<-stand.dat[,-c(25:38,45,46,57,58)]
boxplot(stand44.dat)
# 6.1.kmeans clustering (2000*44)
tstand44<-t(stand44.dat)
set.seed(10)
kmeans.stand44<-kmeans(tstand44,2,iter.max=5)
kmeans.stand44
# 6.2.kmediod clustering-Partitioning Around Medoids(2000*44)
```

```
pam.stand44.e<-pam(tstand44,2,diss=F,metric="euclidean")

summary(pam.stand44.e)

stand44.cor<-cor(stand44.dat,method = "pearson")

stand44.cor.dist<-matrix(nrow=44,ncol=44)

        for(i in 1:44){

        stand44.cor.dist[i,]<-c(1:44)

        for(j in 1:44){

        stand44.cor.dist[i,j]<- 1-stand44.cor[i,j]}}

stand44.cor.dist<-as.dist(stand44.cor.dist,diag=F,)

pam.stand44.cor<-pam(stand44.cor.dist,2,diss=F,cluster.only=F,do.swap=F,keep.diss

        =F,keep.data=F)

summary (pam.stand44.cor)

# 6.3.Hierarchical Cluster Analysis - Euclidean (2000*44)

stand44.e.dist<-dist(tstand44,method="euclidean")

hclust.stand44.e<-hclust(stand44.e.dist,method='complete')

plot(hclust.stand44.e,labels=NULL)

##      Hierarchical Cluster Analysis - Correlation (2000*44)

hclust.stand44.cor<-hclust(stand44.cor.dist,method='complete')

plot(hclust.stand44.cor,labels=NULL)
```

Program four: Classification using 40 training samples vs. 22 testing samples

```
rm(list=ls(all=TRUE)) # clear all previous data

library(MASS)

library(affy)

library(preprocessCore)

# read in the data

colon.dat<-read.table("C:/Users/bo.liu/Desktop/R/I2000.txt",header=FALSE,sep= ' ')

# convert from a list to a matrix format

colon.dat<-data.matrix(colon.dat)

# 1. Classification - Using log-transformed data.(Use 40 samples as training set and

            22 as testing.)

# 1.1.Log-transformed Data

logcolon.dat<-log2(colon.dat)

logcolon.dat<-data.matrix(logcolon.dat)

# 1.2.Linear discriminant analysis using log-transformed data

T <- c(1,3,5,7,9,11,13,15,17,19,21,23,25:38,40,41,44:47,49,52,53,56:59,61)

N <- c(2,4,6,8,10,12,14,16,18,20,22,24,39,42,43,48,50,51,54,55,60,62)

V2001 <- array(0,62); V2001[N] <- 2; V2001[T] <- 1

augmentlog.dat<-rbind(logcolon.dat,V2001)

tauglog<-t(augmentlog.dat)

tauglog<-data.frame(tauglog)
```

```
training<-tauglog[1:40,]

testing<-tauglog[41:62,]

log.lda<- lda(V2001 ~ .,data=training)

log.lda

predict(log.lda, testing)$class

# 1.3.Minimum Sum of Error-Squared using log-transformed data

group <- array(0,62); group[N] <- 1; group[T] <- -1

b<- data.matrix(group)

y0<-array(rep(1,62),dim=c(62,1))

tlog<-t(logcolon.dat)

ylog<-cbind(y0,tlog)

I<-ginv(t(ylog[1:40,])%*%ylog[1:40,])

alog<- I%*%t(ylog[1:40,])%*%b[1:40,]

alog

testing<- ylog[41:62,]

testing%*%alog

# 2. Classification - Using Quantile Normalized Data

# 2.1.Quantile Normalization

quan.dat<-normalize.quantiles(logcolon.dat,copy=TRUE)

# 2.2.Linear discriminant analysis using quantile normalized data
```

```
augmentquan.dat<-rbind(quan.dat,V2001)

taugquan<-t(augmentquan.dat)

taugquan<-data.frame(taugquan)

training<-taugquan[1:40,]

testing<-taugquan[41:62,]

quan.lda<- lda(V2001 ~ .,data=training)

quan.lda

predict(quan.lda, testing)$class

# 2.3.Minimum Sum of Error-Squared using quantile normalized data

tquan<-t(quan.dat)

yquan<-cbind(y0,tquan)

I<-ginv(t(yquan[1:40,])%*%yquan[1:40,])

aquan<- I%*%t(yquan[1:40,])%*%b[1:40,]

aquan

testing<- yquan[41:62,]

testing%*%aquan

# 3. Classification - Using Standardized Data

# 3.1.Standardization (with MAD)

mad.dat<-c(1:2000)

        for (i in 1:2000){
```

```
        mad.dat[i]<-mad(colon.dat[i,], constant=1)}
mean.dat<-c(1:2000)
        for(i in 1:2000){
        mean.dat[i]<-mean(colon.dat[i,])}
stand.dat<-matrix(nrow=2000,ncol=62)
        for(i in 1:2000){
        stand.dat[i,]<-c(1:62)
        for (j in 1:62){
        stand.dat[i,j]<-(colon.dat[i,j]-mean.dat[i])/mad.dat[i]}}
stand.dat<-data.matrix(stand.dat)
# 3.2.Linear discriminant analysis using standardizated data
augmentstand.dat<-rbind(stand.dat,V2001)
taugstand<-t(augmentstand.dat)
taugstand<-data.frame(taugstand)
training<-taugstand[1:40,]
testing<-taugstand[41:62,]
stand.lda<- lda(V2001 ~ .,data=training)
stand.lda
predict(stand.lda, testing)$class
# 3.3.Minimum Sum of Error-Squared using standardizated data
```

```
tstand<-t(stand.dat)

ystand<-cbind(y0,tstand)

I<-ginv(t(ystand[1:40,])%*%ystand[1:40,])

astand<- I%*%t(ystand[1:40,])%*%b[1:40,]

astand

testing<- ystand[41:62,]

testing%*%astand
```

Program five: Classification using 20 training samples vs. 42 testing samples

```
rm(list=ls(all=TRUE)) # clear all previous data

library(MASS)

library(affy)

library(preprocessCore)

# read in the data

colon.dat<-read.table("C:/Users/bo.liu/Desktop/R/I2000.txt",header=FALSE,sep= ' ')

# convert from a list to a matrix format

colon.dat<-data.matrix(colon.dat)

# 1. Classification - Using log-transformed data.(Use 30 samples as training set and

        32 as testing.)

# 1.1.Log-transformed Data

logcolon.dat<-log2(colon.dat)
```

```
logcolon.dat<-data.matrix(logcolon.dat)

# 1.2.Linear discriminant analysis using log-transformed data

T <- c(1,3,5,7,9,11,13,15,17,19,21,23,25:38,40,41,44:47,49,52,53,56:59,61)

N <- c(2,4,6,8,10,12,14,16,18,20,22,24,39,42,43,48,50,51,54,55,60,62)

V2001 <- array(0,62); V2001[N] <- 2; V2001[T] <- 1

augmentlog.dat<-rbind(logcolon.dat,V2001)

tauglog<-t(augmentlog.dat)

tauglog<-data.frame(tauglog)

training<-tauglog[1:20,]

testing<-tauglog[21:62,]

log.lda<- lda(V2001 ~ .,data=training)

log.lda

predict(log.lda, testing)$class

# 1.3.Minimum Sum of Error-Squared using log-transformed data

group <- array(0,62); group[N] <- 1; group[T] <- -1

b<- data.matrix(group)

y0<-array(rep(1,62),dim=c(62,1))

tlog<-t(logcolon.dat)

ylog<-cbind(y0,tlog)

I<-ginv(t(ylog[1:20,])%*%ylog[1:20,])
```

```
alog<- I%*%t(ylog[1:20,])%*%b[1:20,]

alog

testing<- ylog[21:62,]

testing%*%alog

# 2. Classification - Using Quantile Normalized Data

# 2.1.Quantile Normalization

quan.dat<-normalize.quantiles(logcolon.dat,copy=TRUE)

# 2.2.Linear discriminant analysis using quantile normalized data

augmentquan.dat<-rbind(quan.dat,V2001)

taugquan<-t(augmentquan.dat)

taugquan<-data.frame(taugquan)

training<-taugquan[1:20,]

testing<-taugquan[21:62,]

quan.lda<- lda(V2001 ~ .,data=training)

quan.lda

predict(quan.lda, testing)$class

# 2.3.Minimum Sum of Error-Squared using quantile normalized data

tquan<-t(quan.dat)

yquan<-cbind(y0,tquan)

I<-ginv(t(yquan[1:20,])%*%yquan[1:20,])
```

```
aquan<- I%*%t(yquan[1:20,])%*%b[1:20,]

aquan

testing<- yquan[21:62,]

testing%*%aquan

# 3. Classification - Using Standardized Data

# 3.1.Standardization (with MAD)

mad.dat<-c(1:2000)

        for (i in 1:2000){

        mad.dat[i]<-mad(colon.dat[i,], constant=1)}

mean.dat<-c(1:2000)

        for(i in 1:2000){

        mean.dat[i]<-mean(colon.dat[i,])}

stand.dat<-matrix(nrow=2000,ncol=62)

        for(i in 1:2000){

        stand.dat[i,]<-c(1:62)

        for (j in 1:62){

        stand.dat[i,j]<-(colon.dat[i,j]-mean.dat[i])/mad.dat[i]}}

stand.dat<-data.matrix(stand.dat)

# 3.2.Linear discriminant analysis using standardizated data

augmentstand.dat<-rbind(stand.dat,V2001)
```

```
taugstand<-t(augmentstand.dat)

taugstand<-data.frame(taugstand)

training<-taugstand[1:20,]

testing<-taugstand[21:62,]

stand.lda<- lda(V2001 ~ .,data=training)

stand.lda

predict(stand.lda, testing)$class

# 3.3.Minimum Sum of Error-Squared using standardizated data

tstand<-t(stand.dat)

ystand<-cbind(y0,tstand)

I<-ginv(t(ystand[1:20,])%*%ystand[1:20,])

astand<- I%*%t(ystand[1:20,])%*%b[1:20,]

astand

testing<- ystand[21:62,]

testing%*%astand
```