2007

Data Mining and Analysis on Multiple Time Series Object Data

Chunyu Jiang
Wright State University

DATA MINING AND ANALYSIS ON MULTIPLE TIME SERIES OBJECT DATA

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

By

CHUNYU JIANG
M.S., Wright State University, 2003

2007
Wright State University

Wright State University

SCHOOL OF GRADUATE STUDIES

April 20, 2007

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Chunyu Jiang ENTITLED Data Mining and Analysis on Multiple Time Series Object Data BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Guozhu Dong, Ph.D.
Dissertation Director

Thomas A. Sudkamp, Ph.D.
Director, Computer Science and Engineering
Ph.D. Program

Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies

Committee on Final Examination:

Soon Chung, Ph.D.

Ping He, Ph.D.

Saverio Perugini, Ph.D.

Matt Rizki, Ph.D.

ABSTRACT

Jiang, Chunyu, Ph.D., Department of Computer Science and Engineering, Wright State University, 2007. Data Mining and Analysis on Multiple Time Series Object Data.

Huge amount of data is available in our society and the need for turning such data into useful information and knowledge is urgent. Data mining is an important field addressing that need and significant progress has been achieved in the last decade.

In several important application areas, data arises in the format of *Multiple Time Series Object (MTSO)* data, where each data object is an array of time series over a large set of features and each has an associated class or state. Very little research has been conducted towards this kind of data. Examples include computational toxicology, where each data object consists of a set of time series over thousands of genes, and operational stress management, where each data object consists of a set of time series over different measuring points on the human body. The purpose of this dissertation is to conduct a systematic data mining study over microarray time series data, with applications on computational toxicology.

More specifically, we aim to consider several issues: feature selection algorithms for different classification cases, gene markers or feature set selection for toxic chemical

exposure detection, toxic chemical exposure time prediction, wildness concept development and applications, and organizing diversified and parsimonious committee. We will formalize and analyze these research problems, design algorithms to address these problems, and perform experimental evaluations of the proposed algorithms. All these studies are based on microarray time series data set provided by Dr. McDougal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Guozhu Dong, for his support, patience, and encouragement throughout my graduate studies. It is not often that one finds an advisor that always finds the time for listening to the little problems and roadblocks that unavoidably crop up in the course of performing research. His technical and editorial advice was essential to the completion of this dissertation and has taught me innumerable lessons and insights on the workings of academic research in general.

My thanks also go to the members of my major committee, Soon Chung, Ping He, Saverio Perugini, and Matt Rizki for reading drafts of this dissertation and providing valuable comments that improved the presentation and contents of this dissertation. I am also grateful to my lab mates Shihong Mao, Lijun Chen, and Ying Sun for discussion and being supportive. My thanks also go to my friends and my colleagues, Chang Liu, Lena Yemelyanov, Matt Senne, Mudassar Qureshi, Scott Erickson, and Ye Liu, for being supportive and proof reading. Last, but not least, my parents, Jinrong Zhang and Jiali Jiang, and my sister, Xiaoyan Jiang, receive my deepest gratitude and love for their dedication and the many years of support during my studies that provided the foundation for this work.

1. INTRODUCTION

Huge amount of data is available in our society and new data is becoming available often. Previously available data can be used in new areas in other forms and new data sources from new applications emerge quickly as well. How to transform this data into useful information and knowledge has been the major concern in data mining.

The data type we focus on in this dissertation is called *Multiple Time Series Object* (*MTSO* in short) data. Each *MTSO* data can be viewed as a bundle of time series, and it is associated with a class label. The typical examples for *MTSO* can be sensor network data, microarray time series data, and *EEG* (electroencephalogram) data.

- Sensor network data are produced by sensor network from different applications. Each sensor inside the sensor network continuously produces data and hence produces a time series. All these time series data lead to a *MTSO* data.
- Microarray data came into being several years ago and microarray time series data emerged recently. Microarray data has thousands of genes as features. Each gene's expression values vary from time to time which produces a time series. When multiple microarray data is combined together, we have an *MTSO* data. Normally speaking, *MTSO* data has a relatively short time period.
- Electroencephalography is the neurophysiologic measurement of the electrical activity of human brains. Electronic signals are collected by recording from electrodes placed on the scalp or, in special cases, subdurally or in the cerebral

cortex. The resulting traces are known as an electroencephalogram (*EEG*) and represent an electrical signal (postsynaptic potentials) from a large number of neurons. *EEG* data can be used in many areas such as building Brain Computer Interfaces (*BCI*s). Normally speaking, the associated time period is relatively long.

In our research, we will use gene expression data as our data source. Our primary goal is to consider the following issues:

- **MTSO data normalization.** Normalization needs to be done not simply because of the purpose of removing noises. In order to analyze the *MTSO* data, we need to compare features to features. So, as a comparison requirement, we have to make these features comparable to each other.
- **Feature selection for classification.** Choose certain feature set from the feature domain that can be used in regular classifiers. For example, when given a microarray data with time value or an *MTSO* data from a test tissue, the first step is to choose certain features. Then, the classifier should be able to identify the class label for the tissue, cancer or not, controlled or exposed, or even cancer type. We will discuss ranking and classification methods for both microarray data and *MTSO* data.
- **Robust features selection and fragment classification.** Time value is not always available for microarray data. Sometimes, the whole *MTSO* data may only be available partially, e.g. microarray. For toxic chemical exposure detection, when only a part of *MTSO* is provided, we need the classifier to find its class label regardless of the missing time value. We developed the idea of robustness and we

use robust genes to get around time when it is unavailable or inaccurate.

- **Wildness.** There are genes that behave significantly different among different groups: it is expressed very consistently in the normal group because of being regulated and it is expressed extremely wildly in the other group, e.g. after toxic exposure. We developed wildness measurement and used these wild genes in fragment classification as well.
- **Time recovery**. We are going to discuss how to estimate the time value of partial available *MTSO* data. For certain features, their values may be quite different from time to time in a highly regulated manner. Identification of these features' time characteristics will help us to estimate the time value of a microarray data when given a value of such a feature.

1.1. Overview of MTSO Data

Each *MTSO* is in the form of a data matrix and associated with a label. In the matrix, each row of a *MTSO* matrix is a series of real numbers and it represents a time series and it has an attribute such as a sensor or a gene. The number of rows can be large or small and it depends on the nature of the application. The microarray time series data for gene expressions normally has thousands of rows, where each row represents a time series of a gene. The *EEG* data has only tens of rows and each row represents a time series of the sample of the EEG signal recorded by an electrode.

Each column of the matrix represents a snap shot of the test samples and it is associated with a time point. The number of columns can be larger or small and it depends on the amount of time data collection procedure consumes and how many sampling time points are available. The microarray time series data for gene expression normally have just a few time points and the *EEG* series data generally are quite long.

1.1.1. Introduction to Microarray Technology

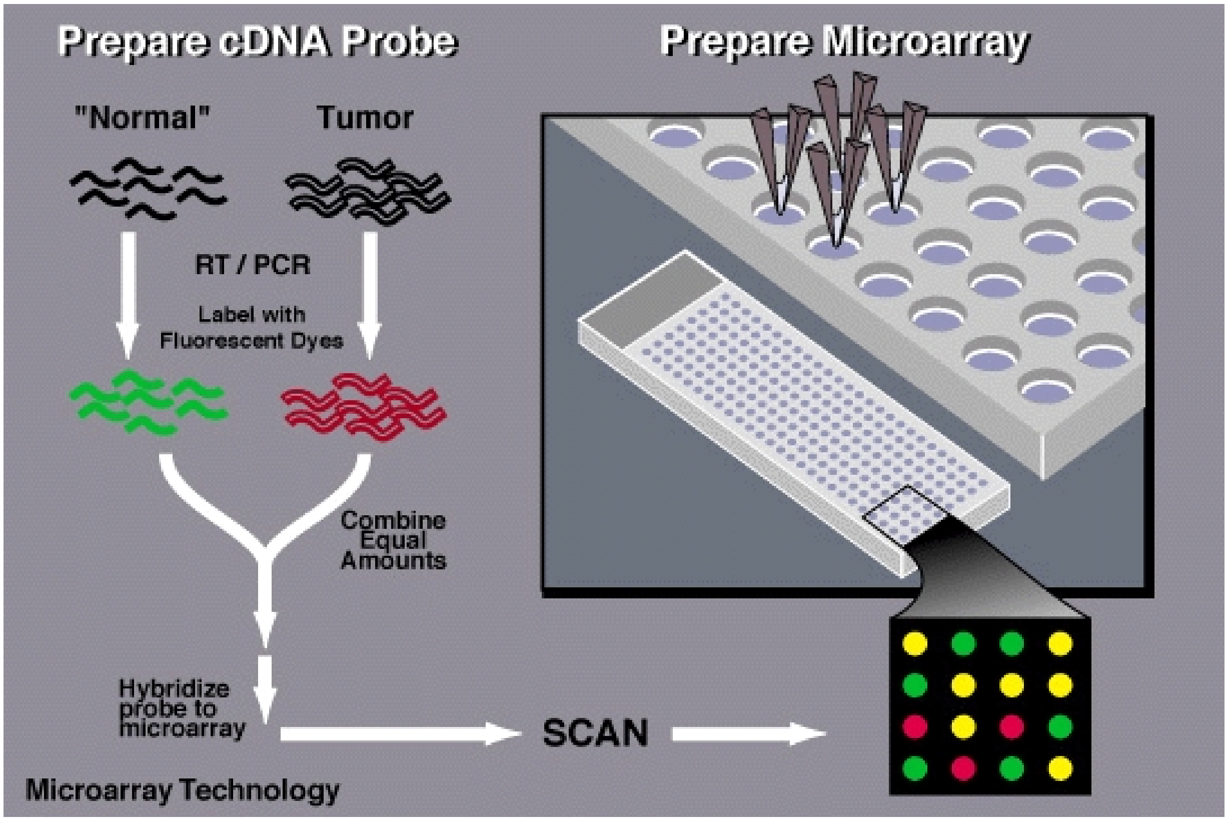

Figure 1.1: Scheme of microarray technology

From http://www.accessexcellence.org/RC/VL/GG/microArray.html

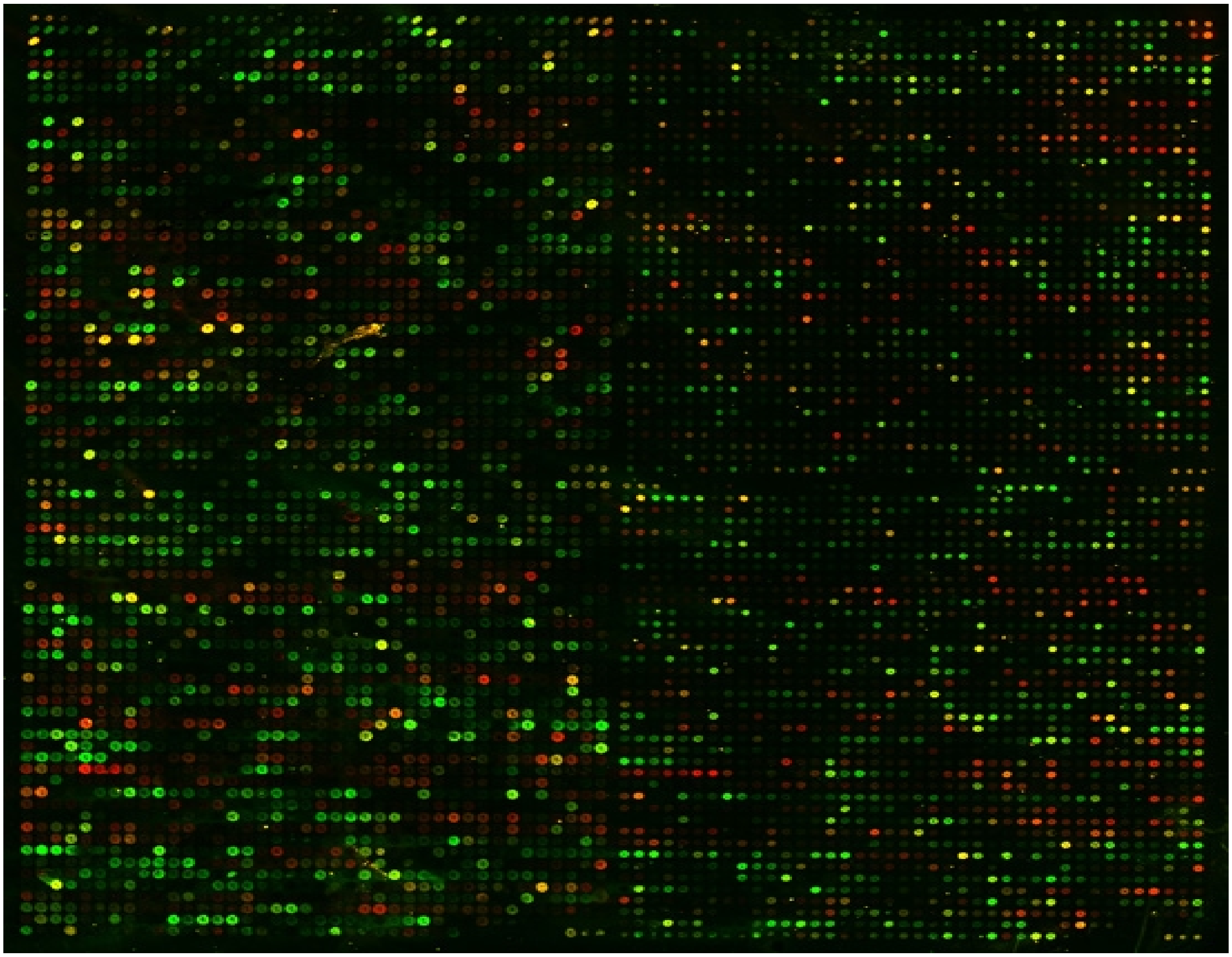

Figure 1.2: Full yeast genome on a chip
From http://cmgm.stanford.edu/pbrown/yeastchip.html

Microarray technology is a relatively new way of studying the working mechanism of large numbers of genes including interaction of these genes with one another and genes working simultaneously. A DNA microarray is also commonly known as gene or genome chip, DNA chip, or gene array. The goal of this technology is to help scientist to gain insights into underlying biological processes. A microarray is typically a glass slide. DNA molecules are attached to spots on these slides. The number of spots can be quite vast.

This technology first applies fluoresecence-labled DNA molecules onto to gene chips. Each spot on the chip can bind to a certain DNA sequence and is associated with a color. Later, lasers are applied to scan these colors, images are stored and the brightness of each spot reveals how much of a specific DNA fragment is present. The brightness value at each spot indicates how active the fragment associated with this spot is.

After the intensity values are converted into numbers, we have a gene expression database including the gene expression data matrix, genes' name, and sample's class label.

Based on DNA microarray technology, there are some other similar technologies developed later on, such as protein microarray technology, gene microarray technology, tissue microarray technology, and et al.

1.2. Organization

In this dissertation, the following topics are discussed in the following order. After briefly describing the *MTSO* data properties, several existing techniques that are used in this project and the related terminologies are briefly introduced in Chapter 2. Literature review is conducted in Chapter 3, where topics such as time series analysis, gene expression data normalization, and cancer classification are discussed. Previous research approaches are also systematically studied. In Chapters 4, 5 and 6, we outline our research problems and we present our approaches together with their experimental results and analysis. These chapters cover the studies we made in topics such as: normalization, GST ranking, ET ranking, *MTSO* classification, fragment classification, wildness concept, robust concept, and time recovery. Then, a review and future work are conducted in Chapter 7. At the end of this dissertation, references are given.

2. PRELIMINARY

In this chapter, we first introduce the *MTSO* data to be studied. Then, some analyzing techniques related to this research are briefly covered. At the end of this chapter, some terminologies and definitions used in this dissertation are given.

2.1. Background of Microarray Data

Time series data is frequently used in statistics and signal processing. It is a sequence of signals that are measured at successive time points for a subject. Each signal is associated with a time point. The intervals may or may not be uniform. Typical examples include electronic signals, stock prices and currency exchange rates.

Microarray data is the data collected by using microarray technology. It is a snap shot of expression values of thousand of genes at a certain time point. All genes have the same time domain and are sampled simultaneously. Microarray is commonly called gene chip, DNA chip, or biochip. Also, microarray data is not limited to DNA microarrays. Other microarray data include protein microarrays, tissue microarrays, transfection microarrays (also called cell microarrays), chemical compound microarrays, and antibody microarrays. Each microarray data has a time value in our research. This value stands for the sampling time of the tissue.

Microarray time series data is a collection of microarray data in the form of a matrix. It is obtained by retrieving microarray data from tissues at several different time points.

Multiple Time Series Object ***(MTSO)*** is a data object that contains several time series. These time series have the exactly same sampling time domain over the same features. In our research, each data object is associated with a class label. The data set we focus on is microarray time series data; the topic of this dissertation is how to analyze microarray time series data.

We hypothesize that useful patterns exist in MTSO. These patterns are more informative and accurate for classification because they are extracted from time series data, as opposed to the data extracted from a single time point. With these patterns and given test samples, we will be able to build more accurate classifiers that give more reliable classification results, revealing more information than previous methods could, and recognize/analyze more patterns. While current research results are not quite satisfactory, these newly found patterns will be of great help in certain research areas especially where data varies frequently and drastically due to time changes, such as cancer classification using microarray time series data.

There are several popular microarray data sets available online that has been studied by other researchers:

- Colon cancer data set. It was originally reported by Alon et al. in [2]. This data set is collected from colon biopsy samples. It has 62 objects that are associated with colon epithelial cells. Among them, 40 biopsies are from tumors (labeled as "positive") and 22 normal (labeled as "negative") biopsies are from healthy parts of the colons of the same patients. Two thousand out of around 6500 genes were selected based on the confidence in the measured expression levels. It is available at http://www.molbio.princeton.edu/colondata.

- Ovarian cancer data set. This data set contains 32 objects: 15 ovary biopsies of ovarian carcinomas and 13 biopsies of normal ovaries, and 4 samples of other tissues. This data set includes approximately 100,000 genes.
- Leukemia data set. This data set is a collection of expression measurements reported in [23]. It has 72 samples: 25 samples of acute myeloid leukemia (AML) and 47 samples of acute lymphoblastic leukemia (ALL). The tissues were taken from 63 bone marrow samples and 9 peripheral blood samples. The data, 72 samples over 7129 genes can be retrieved at http://www.genome.wi.mit.edu/MPR.

2.2. Feature Selection

Feature selection has been the focus of interest for a while and much research work has been done. It tries to choose a subset of features from the feature domain according to certain criteria, such as minimizing subset size without degrading the classification accuracy drastically.

In [14], four major steps were proposed in a typical feature selection method:

1. Generation procedure generates the candidate subset;
2. Evaluation function evaluates the subset;

3. Stopping criterion decides when to stop;
4. Validation procedure checks if the subset is valid.

There are many algorithms available and they can be divided into different categories [14] according to their generation procedures, such as complete, heuristic, random, and evaluation functions, such as distance, information (or uncertainty), dependence, consistency, or classifier error rate.

2.3. Classification

The classification problem has been studied in several areas such as machine learning and database. Many algorithms have been developed such as decision tree, Bayesian network, neural network, and etc.

Cancer classification using gene expression data is special because of the uniqueness of gene expression data. It is a relatively new area and previous studies have shown that the expression values of certain genes are closely related to cancers. Although quite a few methods were developed, comparisons in [36] suggested that no single method is superior over other methods consistently.

The challenges revealed earlier in [36] include: limited amount of available data, inherent noise in the existing data, huge numbers of features and most of them are irrelevant, and limited accuracy problem. But there is another challenge that has not been addressed yet. Microarray data is a time sensitive system. In another word, different features (genes) have different response characteristics; different genes work differently at different time. For toxicity data, it is unrealistic to expect an early response gene to act reliably as an informative gene through out the whole time series. To improve the reliability of classifiers, we need to focus on finding a robust feature set.

2.4. Statistical and Biological Significance

Statistical significance and biological significance are two closely related concepts and they are the key features scientists expect from the cancer classifiers by using microarray technology.

Statistical significance means the given effect is unlikely to have occurred randomly and it has some rules applied. The significance level refers to the randomness level, such as 5% or even less. As the significance level decreases, so does the possibility that the effect occurs by chance. This increases the certainty or precision of the prediction. “Any effect observed in a study or experiment carries with it some degree of uncertainty, or imprecision, because of randomness and variability in most biological phenomena. Values that has a low probability to happen by chance are called statistically significant” – available from http://www.rerf.or.jp/eigo/glossary/stats.htm

Statistical significance may not be informative enough to scientists, and that is why biological significance concept is introduced. For example, a phenomenon such as an increase in pulse of 1 per minute can be statistically significant if tested in a large sample, but it has no practical clinical implication by itself. The concept of biological significance is based on statistical significance. According to the events that are statistically significant, some of them may have strong impact on health. In other words, the observed effect has to be statistically significant if it is biologically significant; on the contrary, an effect can be just statistically significant without being biologically significant. – available from http://www.rerf.or.jp/eigo/glossary/biols.htm.

In order to obtain biological significant results via cancer classification, we need to focus on biological meaningful classifiers who can give out information such as gene markers.

2.5. Terminologies and Definitions

In this section, we are going to define and introduce some terminologies and notations to be used throughout this dissertation.

Using microarray as an example, each *MTSO* consists of a number of microarray snapshots of tissues under similar conditions (exposed or controlled, and coming from a common person or animal), taken at a number of time points over a period of time.

The microarray time series for gene expressions data is a *MTSO*. It has a number of genes $g_1, g_2, ..., g_\gamma$, and a number of time points $t_1, t_2, ..., t_\tau$. Each unit of *MTSO* is a matrix *X*; such a matrix is analogous to a transaction for market basket data or a relational tuple for relational data; *X* has the genes as rows and the time points as columns. *X[g,t]* gives the reading for gene *g* at time *t*.

Data is grouped into classes and each *MTSO* is associated with a class label. For now, we consider the two-class situation, positive or negative. Let $P = \{P_1, ..., P_\mu\}$ be a set of positive (e.g. exposed) *MTSO*s, and $N = \{N_1, ..., N_\nu\}$ (μ and ν don't have to be equal) be a set of negative (e.g. control) *MTSO*s. Here, $P_1[g,t]$ refers to the gene expression value of gene *g* at time *t* of a positive *MTSO* P_1; $P_1[g]$ stands for the time series for gene *g*; $P_1[t]$ stands for the snapshots of for all genes at time point *t*.

3. LITERATURE REVIEW

Much research work has been done on several topics such as similarity search in time series analysis, microarray gene expression data normalization and classification, and microarray time-series data analysis. These topics are discussed in the following sections.

3.1. Time Series Analysis

Much research has been done on single time series data analysis. Research has been focused on searching similarity among time series. Normally, these algorithms can be divided into three parts:

- A representation technique which abstracts the presented time series data.
- A distance measure between two given time series data.
- An efficient indexing method.

Two categories exist in sequence similarity matching:

– Whole matching: to measure the similarity between two whole time series.

– Subsequence matching: query sequence is smaller. To measure the similarity between a short sequence and a subsequence of a time series.

In studying time series similarity matching, most research work focuses on how to index time series data for efficient future similarity search. Several basic indexing requirements are:

- Fast.
- Correct.
- Small space overhead.
- Dynamic: easy to insert, delete, and append sequences.
- Flexible length.

Locating and retrieving data efficiently is hard and normally transforms are necessary for indexing. Two kinds of transforms, data-dependent are data-independent transforms, were developed and there are advantages and disadvantages for both methods. Data-dependent transforms are fast but they may need to re-compute the transformation matrix when algorithms are applied to different data set [3].

Several key issues in time series transforms are: to efficiently squeeze the data volume, to keep as much of the original information in transformed results as possible, to preserve the Euclidean distance between time series by using orthonormal transforms.

Many transform or compression techniques were developed in [3], [21], [29], [27], [28], [31], [37] and [1], such as DFT (Discrete Fourier Transformation), DWT (Discrete Wavelet Transformation), SVD (Singular Value Decompression), SVDD (Enhanced Singular Value Decompression), PAA (Piecewise Aggregate Approximation), APCA (Adaptive Piecewise Constant Approximation), SAX (Symbolic Aggregate approXimation), and SDA (Shape Description Alphabet) etc. These techniques can be divided into two categories: mathematical transforms such as Discrete Fourier

Transformation and approximation techniques such as Shape Description Alphabet.

Comparisons are made between the two most popular techniques in [10] and [54]: DFT and DWT. Different conclusions are drawn as to which method is superior.

The differences between DWT and DFT include:

- DFT maps a one dimensional time domain discrete function into a representation in frequency domain.
- DWT maps it into a representation that allows localization in both time and frequency domains.
- DWT is more appropriate than DFT in time series because it reduce the error of distance estimates on the transformed domains.

Both of them share some similarities such as:

- Keep the first several coefficients to approximately keep most energy.
- First DWT/DFT coefficients are the same because they represent the average value.
- DFT coefficients spread suggests that most energy associates with low or high frequency, not middle level.
- DWT coefficients spread suggests that most energy is associated with low frequency.

Tests were done to verify the hypothesis made in [10]. Testing results show that DWT is superior to DFT if mirror effect is not considered. But it doesn’t hold otherwise.

3.2. Gene Expression Data Analysis

Microarray gene expression data provided by biological scientists has become available recently. Microarray technology makes uses of the results of the genome projects. This technology tries to answer questions about what expression value of what gene in which kind of cells in what kind of organism at what time under what conditions. Different names are given like DNA microarrays, DNA arrays, DNA chips, gene chips. Interesting topics include normalization and classification.

3.2.1. Normalization

As mentioned in [42], the hypothesis of microarray analysis is: “the measured intensities for each arrayed gene represent its relative expression level.” Normally speaking, patterns that are biologically meaningful are discovered by comparing measured expression levels between different states on a gene-by-gene basis. In order to achieve this purpose, transformations have to be done to the original data, so that these gene expression levels are comparable to one another, since raw data is generally obtained by low-quality measurements.

Notably, in research, normalization is the first transformation step in microarray data analysis. It is a crucial step because systematic variations exist when biologists obtain data through experiments. It adjusts each gene’s hybridization intensities to balance them so that meaningful biological comparisons are possible. The sources could be: differences in labeling or detection efficiencies of various fluorescent dyes, laser power differences, experimental errors, unequal quantities of starting RNA, and system biases.

There are basically two ways to perform data normalization: per chip normalization or per gene normalization. Per chip normalization has been studied extensively and per gene normalization is a relatively new research field.

Per-chip normalization is a scaling method that tries to adjust the average expression value obtained from each gene chip to the same level. When applied to the *MTSO* data, per-chip normalization is used to adjust the average value of each column to approximately the same value when differences exist in probe preparation, hybridization conditions, etc. This helps to eliminate the bias among different samples and makes all the samples comparable. This method works for similar samples and the disadvantage is that it cannot detect outliers. Several popular methods are available:

Per-gene normalization is another normalization method that compares the values of each gene across all the samples. When applied to the *MTSO* data, per-gene normalization is to make values at each row at the same range. It is helpful when gene-to-gene comparisons are necessary especially when their original values are of different level but certain patterns may exist. It has not been as extensively explored as per-chip normalization. The reason is that people have not started to analyze *MTSO* data yet.

Four categories of per-chip normalization methods are listed below together with some individual algorithms.

- **Housekeeping genes:** A set of genes that are expressed in such a way that little variance is involved. The scaling factor can be easily calculated by comparing different expression values of these housekeeping genes in various samples. This method differs from other methods; where synthetic references, such as using the average value of the whole microarray set, must be calculated.
- **Internal reference:** algorithms in this category are based on internal standard genes or set of internal standard genes. One or more genes are chosen from the genes being studied when scaling is unnecessary.
- **Internal globalization:** Sum, mean, median, quantile/percentile, trimmed mean, asymmetric trimmed mean, linear regression (centralization).
- **External reference:** External standard genes are used to remove noises and this method is not as popularly adopted as other approaches.

In [34], a complete comparison was made for most available normalization methods at that time such as:

- Centralization in [58]. This method has two major steps to compensate the commonly believed fact that gene expression values from different samples are approximately proportional to the other samples. This method calculates a so-called normalization factor. First, the quotient of the constants of proportionality is estimated for each pair of microarray data. Next, an optimally consistent scaling of the microarray data is calculated based on the matrix of the pair wise quotients resulted from the previous step.

- In [17], authors produced two synthetic poly(A)-RNAs that were generated by PCR and in vitro transcription. In their algorithms, these two synthetic poly(A)-RNAs were used as external standards for normalization when internal reference were not available.
- After studying [5], [30], [48], [55], [52], and [15], Table 3.1 from [34] is provided to show the basic formulas for each algorithm from these papers.

Methods	Formula of normalizing factor
Internal standard genes	$\eta = S_{ref}$
Set of internal standard genes	$\eta = \overline{S}_{ref} = \frac{1}{N_{ref}} \sum_{i_{ref}=1}^{N_{ref}} S_{i_{ref}}$
Sum	$\eta = \sigma = \sum_{i=1}^{N} S_i$
Mean	$\eta = \overline{S} = \frac{1}{N} \sum_{i=1}^{N} S_i$
Median	For odd N … $\eta = \hat{S} = S_{r=(N+1)/2}$ For even N … $\eta = \hat{S} = \frac{1}{2}(S_{r=N/2} + S_{r=N/2+1})$
Quantile/percentile	$\eta = {}^{q}\hat{S} = S_{r=round(q*N)}$ $\quad 0 < q < 1$ $\eta = {}^{p}\hat{S} = S_{r=round\,(p/100*N)}$ $\quad 0\% < p < 100\%$
Trimmed mean	$\eta = {}^{p}\overline{S} = \frac{1}{round((1-p)N)} \sum_{r=round(p/2)}^{round(N(1-p/2))} S_r$
Asymmetric trimmed mean	$\eta = {}_{l}^{h}\overline{S} = \frac{1}{round((1-h-l)N)} \sum_{r=round(h)}^{round(N(1-l))} S_r$

N, total number of genes in the data set.
i, index of genes.
r, rank of genes.
s_i, background corrected signal of gene with index i.
s_r, background corrected signal of gene with rank r.
N_{ref}, number of reference genes.
i_{ref}, index of reference genes.
η, scaling factor to be applied.

Table 3.1: Formulas of some normalization methods

3.2.2. Cancer Classification

Cancer classification is a hard topic and it has been studied extensively. Cancers that have similar symptoms can be totally different and even different subtypes can result in significantly different responses to similar therapy. The classification has relied on specific biological insights in the history and still needs to be solved. A concrete example is: Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML) have major difference in treatment. Even the distinction between AML and ALL has been well established, no single test that can establish diagnosis. As a result, systematic and unbiased approaches to identify cancer types have always been the research target.

Classification based on gene expression data is a promising research topic because some studies show the feasibility of cancer classification based solely on gene expression monitoring. As noted in [36], previous research work is limited in clinics and the reliability of classification results is limited as well. But the invention of gene chips and microarray data stimulated its development because some cancers can be reflected by the certain genes' expression values. Because microarray data provides the ability to simultaneously monitor numerous genes' expression values, using microarray data in cancer classification is promising and a number of classification methods have been proposed. A good classification method may provide statistical significance as well as biological significance. In this section, several existing classification algorithms are reviewed and a complete comparison is given in Table 3.2.

There are some special characteristics on gene expression data that make the research work unique.

- High dimensionality and limited sample sizes.
- Existing noises: biological or technical.
- Irrelevance of most genes to cancers.
- Availability of data sets is limited.

In [23] and [50], a generic approach to cancer classification based on gene expression monitoring by DNA microarrays was developed. This method (referred to as *GS method* later) is applied to human acute leukemia as a test case in order to distinguish acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). Several issues are discussed in the paper. These issues are: how to determine if the genes necessary to make a predication are available; how to use these genes to predict, and how to test the validity of class predictors. Genes were sorted according to their degree of correlation. They use "neighborhood analysis" to find the informative genes (genes that are highly correlated with the class labels, which means to a gene that is uniformly expressed high in one class and uniformly low in the other) and they are used in the predictors in weighted voting manner. In this method, a ratio called *SNR* (stands for Signal-To-Noise, $P(g,c)=\frac{\mu_1(g)-\mu_2(g)}{\sigma_1(g)+\sigma_2(g)}$, where $\mu_1(g),\mu_2(g)$ and $\sigma_1(g),\sigma_2(g)$ stands for the mean and standard deviation of the $\log_{10}$ of the expression values of gene *g* in class 1 and 2 respectively.) is used to measure the correlations between class and gene expression value. The key idea is to select genes with expression values that are significantly different between classes but quite uniform inside one class. After feature selection, each informative gene casts a vote towards one of the two classes in a weighted manner. The votes were summed and the sample was assigned to the winning class if prediction strength exceeded a predetermined threshold. Cross-validation method is used to test the accuracy of the predictors. The results show the feasibility of cancer classification by using gene expression data.

In [32], authors present a Naïve Bayesian method for classifying tissues via using DNA microarray data as well as a gene selection scheme explicitly designed to optimize the classification algorithm. In this method, each gene from each class is modeled as a Gaussian distribution. Authors use a likelihood-based metric to select the most useful subset of genes in the classifier and use Naïve Bayesian classifier in prediction. Independence is assumed and the Naïve Bayesian method is used to assign class labels to a given sample. Because the gene selection and classification steps are orthogonal to each other, authors combined 2 possible gene selection methods (likelihood-based or GS) and classification (Naïve Bayesian or G-S) and performances are compared among the four possible combinations. The results show that Naïve Bayesian classifier and the G-S performed better on different data sets, no one is superior all the time. The results also show the direct correlation between the performance of the NB classifier and the magnitude of the Likelihood scores on the selected genes, which can be used to give suggestions that whether the Naïve Bayesian method should be adopted. The authors also pointed out that Naïve Bayesian method can be easily extended from bi-class classification to multi-class cases.

Artificial Neural Networks (ANNs) is an abstract simulation of a real human brain. It is composed of a certain numbers of neurons. This simulation can accomplish a set of tasks such as pattern recognition and classification after being trained. The training procedure is normally done by calibrating the parameter of the ANN by literately minimizing errors. Authors in [33] adopted artificial neural networks model and try to develop a classifier for cancers via only the gene expression values and the small, round blue-cell tumors of childhood (SRBCTs) are chosen as the training data set. Three major steps are developed on these 4 categories data.

– Principle component analysis to reduce the data dimensionality.

– Relevant gene selection.

– ANN prediction.

Because of the limited data availability and the achieved performance, only linear model is adopted. Still, the results show very good performance and no sign of over-training. The authors also calculated the sensitivity of the classification by changing the expression value of each gene. Authors also produced a list of genes sorted by their significance on classification.

Authors in [59] tried the classification tree approach and proved to be significantly more accurate for discriminating among distinct colon cancer tissues than other statistical approaches. Recursive partitioning technique is used to classify tissues of gene expression data. Recursive partitioning is a classification technique that predicts the class label of a given test object based on feature information. The difference between Fisher's linear discriminant analysis and recursive partitioning technique is: Fisher's linear discriminant analysis uses linear combinations of the covariates while the recursive partitioning technique extracts homogeneous features from the data. In this paper, a binary decision tree is constructed through recursively partitioning and finding the best gene to divide sample data group into smaller sample data groups in each loop. In splitting the nodes at each internal node of the tree, an entropy function, $P\log(P)+(1-P)\log(1-P)$, is designed to evaluate the purity of each sample data group. Furthermore, a pruning procedure is considered to cut off redundant groups to avoid overgrowing. Also, to accurately evaluate the tree quality, cross-validation is adopted. Colon cancer data retrieved from http://www.sph.uth.tmc.edu/yhgc is used in the experiments. The results show that using only three genes, IL-8 (M26383), CANX (R15447), and RAB3B (M28214) is sufficient to achieve a very satisfactory classification accuracy (98%).

The authors of [4] rigorously assessed the possibility of classification approaches via using gene expression data. A novel cluster based classification methodology is presented and is compared with two other approaches, Boosting and Support Vector Machines on three data sets: colon cancer data, ovarian cancer data, and leukemia data. Leave One Out Cross Validation (LOOCV) is used to evaluate the prediction level of the methods mentioned above.

The authors of [4] compared two classification methods based on similarity. One method is based on nearest neighbor methods and it is used as a strawman in this paper; and a more sophisticated classification method is put forward as an improved version of *CAST*. Cast is based on clustering method and is implemented in the BioClust analysis software package.

The *CAST* algorithm utilizes a threshold parameter to control the granularity of the resulting cluster structure. *CAST* builds one cluster at a time until all objects are assigned to clusters. For the clustering results, objects must have high similarity to the cluster to which they are assigned; and have low similarity to other clusters. This threshold has a significant effect on the number of clusters in the results: the higher the threshold is, the smaller the size of a cluster is and the more the number of clusters is.

In order to find the most appropriate value for this threshold, a measurement is proposed to evaluate the compatibility of cluster structure and the label assignment. This measure is called matching coefficient [16]. It favors uniformly labeled clusters and panelizes situations such as numbers of clusters have the same label.

[4] also developed other classifiers such as large margin classifiers via which the decision surface between different classes are studied directly. Two methods are considered: support vector machine (SVM) and boosting.

[41] designed a new approach that combines available methods, SVM and GA [40], together to achieve a satisfactory classification result.

A review of these methods is provided by [36] and Table 3.2 from [36] is listed below to compare different classification methods.

	Multiple class	Strategy	Biologically Meaningful	Scalability
SVM	No	Max-margin	No	Good
Boosting (decision-tree)	Yes	Max-margin	Yes	Classifier dependent
Decision tree	Yes	Entropy function	Yes	Good
KNN (k<=1)	Yes	Similarity	No	Not scalable
CAST	Yes	Similarity	No	Not scalable
GS	No	Weighted voting	Yes (gene selection)	Fair
FLDA	Yes	Discriminant analysis	No	Fair
Neural network	Yes	Perceptrons	No	Fair
Naïve Bayes	Yes	Distribution modeling	No	Fair

Table 3.2: Cancer classification methods review

3.3. Microarray Time Series Data Analysis

As pointed out in [9]: “DNA replication, chromosome segregation, and mitosis define a fundamental periodicity in the eukaryotic cell cycle. Precise coordination of the unidirectional transitions between these stages is critical to cell integrity and survival.” In other words, the normal cell activity depends on the appropriate cell cycle regulation. And scientists expect to find irregular regulations for abnormal cells, such as cancers.

The regulation has been studied by scientists separately before microarray technique became available. Microarray time series data is comprised of thousands of time series data, or in other words, tens of microarray data. With this technique, one can easily monitor numbers of genes simultaneously to help scientists reveal novel functional and physical organization in coordinate gene regulation. This makes more comprehensive study possible. The most commonly used data set is called Cho/Spellman Yeast data set obtained by Cho [9] and Spellman [49], which are available at http://genome-www.stanford.edu/cellcycle. The data set is comprised of four time-series data sets. Each of them contains temporal concentration measurements for ORFs in yeast. And the starting stage of these cells is synchronized so that the cells are approximately in the same state. The time-series courses are repeated through more than one period for Elu, more than two periods for alpha and cdc28, and more than three periods for cdc15.

Certain research has been done based on that data set. The focus of that research was on finding the regulatory relationships, such as activation and inhibition, among different genes. As mentioned in [22], previous research's performance is not good enough. They tried to implement previous algorithms, such as time series analysis, to check the known regulations. Unfortunately, less than 20% of the known regulator pairs exhibited strong correlations, which is neither helpful nor convincing in finding the pathways.

They designed an edge detection function which they claimed to be significantly better in finding regulatory candidates. The edge detector favors similarity of local signals on the curves and acts as a conservative and biologically significant filter in order to maximally eliminate meaningless information. It represents each gene as an array of quadrary edges that only biologically significant and reliable expression level changes are involved. Then a score function is designed for each pair of gene edges. As for the output of their algorithm, results show that this algorithm recognizes certain interesting putative pairs missed by other methods,

The authors also implement integrated analysis by interleaving the 4 time-series curves for each gene because they believe that integrated analysis of the Cho/Spellman data sets is more informative than analyzing each data set separately.

In [57], another approach using dominant spectral component is proposed. The assumption of this method is: regulatory gene pairs vary periodically at a constant and relatively similar frequency. In this method, each time series expression sequence is decomposed into a set of discrete-time damped sinusoids of different frequencies. Parameters in this model, such as amplitude, damping factor, normalized frequency and phase angle are determined based on the autoregressive model commonly used in signal processing. Correlation of two time sequences is reformulated as a sum of scaled sub-correlations. It is claimed that this method is superior in dealing with time delays and many of regulatory genes pairs missed by traditional correlation method can be identified.

3.4. Multiple-Class Classification

Bi-class classification is the most common technique to be considered or studied. Numbers of classifiers are built to distinguish two groups of objects. A discriminant function is designed to separate these two groups of objects as clearly as possible.

Although bi-class classifications are important, multiple-class classifications are more practical. A concrete example is: recognize more than one kind of toxins in the microarray experiments. But, techniques used in bi-class classification might not be appropriate or fit in naturally in multi-class classification problem.

As pointed out by Friedman in [19], multi-class classifications are more difficult than bi-class classification because more class boundaries need to be considered.

Friedman suggested a very intuitive approach for the K-class problem:

- Solve each of the two-class problems
- For a test object, combine all the pairwise decisions to form a K-class decision by assigning the test object to the class that wins the most pairwise comparisons.

Friedman points out that this rule is equivalent to the Bayesian rule when the class posterior probabilities p_i (at the test point) are known:

$$\arg\max_i [p_i] = \arg\max_i [\sum_{j \neq i} I(p_i/(p_i + p_j) > p_j/(p_i + p_j))]$$

Friedman's rule only requires an estimate of each pairwise binary decision. In many cases, not only simple classification results are available, but also the estimated class probabilities. When these estimation results are present, Friedman's rule can be improved.

In [26], authors discuss a strategy for polychotomous classification: firstly classify the given objects pair wisely, and then coupling the estimates together, which is similar to the Bradley–Terry method. The nature of the class probability estimation is studied and the performance is examined. Classifiers such as linear discriminants, nearest neighbors, adaptive nonlinear methods and the support vector machine are studied.

The following question was discussed in that paper: given a set of mutually exclusive events *A*$_1$, *A*$_2$, ... *A*$_k$, as well as pair-wise probabilities, $r_{ij} = \Pr ob\{A_i \mid A_i \bigcup A_j\}$, we need to find a set of probabilities $p_i = \Pr ob\{A_i\}$? The authors are trying to find the best approximation in model $u_{ij} = \dfrac{p_i}{p_i + p_j}$, or an equivalent model $\log u_{ij} = \log p_i - \log(p_i + p_j)$, where u_{ij} is the expected value of r_{ij}. Generally speaking, solutions may not exist for the given question because of the lack of equations comparing to the independent parameters.

The optimized solution can be achieved by maximizing a Kullback-Leibler distance criterion between u_{ij} (the expected value of r_{ij}) and r_{ij} together with Pairwise threshold optimization in order to improve the algorithm complexity,

$$l(p) = \sum_{i<j} n_{ij} [r_{ij} \log \frac{r_{ij}}{u_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - u_{ij}}]$$

On most occasions in the experiments, the pair-wise procedures perform better than linear discriminant analysis in most occasions. Threshold optimization improves performance for Friedman's max rule and the coupling rule. And quadratic discriminant analysis and pairwise coupling performs equally.

In Bayesian classifiers, the probability density function is estimated and the class label is assigned according to the highest posterior probability. This can be easily extended to multi-class problem. Only extra computation is needed and new objects are still assigned to classes according the highest posterior probability.

For classifiers such as linear discriminant, perceptron, and support vector machine, the fitting procedure of discriminant function from bi-class to multi-class is not straightforward. Several strategies are available:

- Voting algorithms. Organize a classifier committee and each classifier votes for or against a class. The final classifications result is made according to the majority votes.
- Couple a probability number with previous bi-class classification results. The multi-class classifications result goes to the class that has the maximal classification output.

In [51], authors talk about the generalization from bi-class classification to multi-class classification. This paper compared two approaches: voting mechanism and combinations of approximate posterior probabilities.

In voting approach, 2 training possibilities are considered: 1 against rest (1-r) and 1 against 1 (1-1). These two methods suffered from questions such as: rejected by all classifiers, accepted by multiple classifiers, ties in voting. To eliminate these issues, simple methods such as assign these objects to the class with largest prior probability are designed by the authors.

In posterior probabilities approach, a confidence value for each classification is used to avoid inconsequent labeling. Then, maximum rules are applied to combine all estimation results.

Experiments are done using 5 data sets and 3 kinds of classifiers: normal density based linear classifier (LDA), Fisher linear discriminant, and the linear support vector classifier (SVC). LDA belongs to the voting approach and the other two belong to posterior probabilities approach.

Conclusions are drawn from the experiments that:

- For voting approaches, rejections happen frequently in vote 1-r, and vote 1-1 has a relatively low rejection rate; the performance on the not rejected data is normally much better; the results of random assignment to the data being rejected are very poor; vote 1-1 is worse than vote 1-r if rejection is considered, but better than vote 1-r with random assignment of objects being rejected.
- For posterior probabilities approach, better results appear for the 1-r case. When rejection is consider, the performance is at the same level as vote 1-r; otherwise, it is consistently better than vote 1-r with random assignment of objects being rejected.

3.5. Decision Committee Learning

Decision committee learning has been considered to be an effective method in reducing the classification errors. All committee members are given the same input test data and their output results are combined in a certain way in order to give a more reliable and dependable classification decision.

There are two popular decision committee learning approaches: Boosting and Bagging [47] [18] [8] [6].

- Boosting is a method that tries to increase the accuracy of any given learning algorithm. It changes the distribution of the training set based on its performance. Boosting is done in the following steps. It maintains a set of weights over the training set. At the very beginning, all the weights are assigned with the values. But, the values are being changed after each round. In each round, a weak learner is trained with an unsolved problem. The output of the weak learner is then added to the learned function, with evaluated strength that corresponds to the performance. Then, the weights are reassigned.
- Bagging (also called Bootstrap Aggregating) first generating new dataset of the same size as the given dataset. In the new dataset, replicates exist. Then, classifiers are trained according to the new datasets and obviously, these new classifiers are not going to be accurate individually. The final results are achieved by averaging the output or voting. This algorithm is quite satisfactory and it also reduces variance and helps to avoid overfitting. Experimental results and theoritical analysis show that bagging is effective on unstable learning algorithms and will downgrade the performance of stable learning algorithms. In this case, "unstable" means that small changes in the input test data will cause large changes in results.

There are also different ways in combining the classification results from these committee members.

- Voting is a quite straightforward way. The majority of classification result from the available classifiers is considered as the final conclusion. Some methods consider assigning weights to each of the classifiers and drawing the final conclusion after adding them together. The major problem with voting is that it is incapable to take local expertise into account.
- Instead of voting, dynamic integration of classifiers can be very effective. It has the assumption that each member of the committee has some expertise on some sub-areas in the feature space.

Combining classifiers has been established as an important research area. The key to make more accurate classifiers is how to choose a diversified classifier set. In [35], authors discussed the concept of "diversity" in great details in the field of combining binary classifiers. Ten statistics which can measure diversity are studied: four averaged pair-wise measures (the Q statistic, the correlation, the disagreement and the double fault) and six non-pairwise measures (the entropy of the votes, the difficulty index, the Kohavi-Wolpert variance, the interrater agreement, the generalized diversity, and the coincident failure diversity). The relationships between diversity and accuracy are also studied, different conclusion were drawn that no clear relationship is found in this research.

4. MTSO NORMALIZATION AND GST RANKING

This chapter discusses some basic concepts and techniques, such as data normalization, revised GS ranking algorithm, and a microarray classification approach. Following chapters will be based on them.

4.1. Normalization

4.1.1. Motivation

As discussed in previous chapters, there are inevitable variations in the microarray data which make normalization an important pre-process in biological research. Normalization helps researchers to mitigate or totally remove the hazards caused by these variations. There are several techniques which can be used separately or together in normalization.

- Multiply genes' expression values from each array by a constant value to make the mean intensity of each chip the same.
- Adjust the chips by using housekeeping genes that are consistently expressed across all of the samples.
- Match the percentiles of each array.
- Adjust using a nonlinear smoothing curve.

In our research, data normalization becomes even more important. This is because we also need to bring the expression values from different sampling times for different genes into the same range so that different genes are comparable. The methods mentioned above are not suitable for our case, because they were designed for data that either does not have time feature association or does not have a class label. In our research, we consider feature selections on *MTSO* data, which has both time variables and class labels. We need a new approach.

4.1.2. Our Approach

As discussed in chapter 3, previous normalization methods only consider microarray data normalization for an individual microarray. Apparently, we need some new approaches to handle the addition of the time dimension. The major challenge is: according to the nature of genes, they are expressed by vastly differing values, which makes direct comparisons between genes difficult. For example, in our sample data set, gene AA684929_f_at is regularly expressed at about 1000; on the contrary, gene Z49858_at normally has a value of less than 10. No comparison among genes can be made without first bringing them into a comparable level. In order to normalize the whole data matrix, two steps are performed in our normalization procedure:

- **Per chip normalization** is based on the mean value of each chip. All the values on one chip are normalized by dividing them with the mean value of the chip. The rationale for this normalization is that most gene expression values don't change among different classes and the mean value from each chip will remain roughly the same [34]. The primary goal of this step is to remove the differences and noises among different chips as much as possible.

Formally, for each chip *j*, we first compute the mean value for the entire chip: $\eta_j = \overline{S}_j = \frac{1}{n}\sum_{i=1}^{n} S_{i,j}$. (Here, *n* stands for the number of available genes on the chip and *S* denotes the array for the original gene expression values). Then, for every gene expression value, $S_{i,j}$, let the new normalized value be $SS_{i,j} = \frac{S_{i,j}}{\eta_j}$. After this transformation, the values for the same gene coming from different chips are comparable.

- **Per gene normalization** is performed to adjust each gene's value into approximately the same range. Different genes are normally expressed in different ranges. This makes it difficult to compare them directly using their original values. An efficient approach is to scale all the genes' expression values into the same range, such as [0, 1]. Per gene normalization will be done using the following formula.

$$GS_{i,j} = \frac{SS_{i,j} - \min_{k=1}^{m} SS_{i,k}}{\max_{k=1}^{m} SS_{i,k} - \min_{k=1}^{m} SS_{i,k}}$$

  Here, $GS_{i,j}$ is the expression value of gene *i* on chip *j* after the normalization and *m* is the total number of gene chips.

Let Γ be the total number of available time points (microarray sampling time) and *n* be the total number of available genes. The original and normalized *MTSO* data matrices are shown in table 4.1 and 4.2. In such a matrix, a row *i*, $[GS_{i,1}, GS_{i,2}, \ldots, GS_{i,\tau}]$, is a time series of gene *i*; a column *j*, $[GS_{1,j}, GS_{2,j}, \ldots, GS_{n,j}]$, is a microarray data sampled at time point t_j.

	t₁	**t₂**	**…**	**…**	**t τ**
Gene 1	$S_{1,1}$	$S_{1,2}$	…	…	$S_{1,\tau}$
Gene 2	$S_{2,1}$	$S_{2,2}$	…	…	$S_{2,\tau}$
…	…	…	…	…	…
…	…	…	…	…	…
Gene n	$S_{n,1}$	$S_{n,2}$	…	…	$S_{n,\tau}$

Table 4.1: Original *MTSO* matrix.

	t₁	**t₂**	**…**	**…**	**t τ**
Gene 1	$GS_{1,1}$	$GS_{1,2}$	…	…	$GS_{1,\tau}$
Gene 2	$GS_{2,1}$	$GS_{2,2}$	…	…	$GS_{2,\tau}$
…	…	…	…	…	…
…	…	…	…	…	…
Gene n	$GS_{n,1}$	$GS_{n,2}$	…	…	$GS_{n,\tau}$

Table 4.2: Normalized *MTSO* matrix.

4.2. GS Ranking Method with Time Factor: GST

Roughly speaking, this section is concerned with evaluating the performance of a ranking algorithm for microarray data, and making preparations for our methods in the rest of this dissertation (such as time series ranking and robust feature selection). To be more specific, we will first discuss the GS method and its limitations. After showing that this ranking should be time dependant, we will do minor improvements to the GS method and use the improved method for feature selection in microarray data. We will incorporate the ranking results in classifications. Finally, we conduct experiments to evaluate the performance of the improved ranking method. The experiments show that the ranking results are time sensitive, which motivates us for the study on whole-time-series based ranking and robust feature selection.

4.2.1. Motivation

As mentioned in chapter 3, the GS method introduced in [23] is a generic approach to cancer classification based on gene expression monitored by DNA microarrays. In [23], genes were sorted according to their degree of correlation between different classes. That article used "neighborhood analysis" to find the informative genes (genes that are highly correlated with the class labels, meaning that such a gene is uniformly expressed high in one class and uniformly low in the other class); these genes are used in the predictors in weighted voting manner. In this method, a ratio called *SNR* is used to measure the correlations between class label and gene expression value. (*SNR* stands for Signal to Noise Ratio, $P(g,c) = \frac{\mu_1(g) - \mu_2(g)}{\sigma_1(g) + \sigma_2(g)}$, where $\mu_1(g), \mu_2(g)$ and $\sigma_1(g), \sigma_2(g)$ stand for the mean and standard deviation of the $\log_{10}$ of the expression values of gene *g* in class 1 and 2 respectively. The use of $\log_{10}$ is a simple method of normalization so that different genes become more comparable.) Clearly, this method has nothing to do with the time factor. In other words, this technique ignored the time value of the microarray data.

In order to create more accurate and more informative classification results when time is present, we introduce the GST method, which is an enhanced GS method by taking the time factor into consideration and incorporate the GST results into classifiers.

4.2.2. Approach and Application

4.2.2.1. GST Ranking

In the GST ranking approach we divide the microarray data into groups according to their time value and we only apply our GST ranking method inside each microarray data group. Specifically, for a given time point *t*, let $GST(g,t) = \left|\frac{\mu_1(g,t) - \mu_2(g,t)}{\sigma_1(g,t) + \sigma_2(g,t)}\right|$, where $\mu_1(g,t), \mu_2(g,t)$ and $\sigma_1(g,t), \sigma_2(g,t)$ stand for the mean and standard deviation of normalized expression values of gene *g* at time point *t* in class 1 and 2 respectively.

Even though both *GS* and *GST* methods use the SNR formula to rank the genes, there are some differences between the original *GS* method and our *GST* ranking method: *GST* considers one time point at a time, and *GST* uses the mean and standard deviation of normalized expression values of gene *g*, instead of the log_{10} of the original gene expression values as was done in the *GS* method.

	group$_1$	**group$_2$**	**…**	**…**	**group $_\tau$**
	t$_1$	**t$_2$**	**…**	**…**	**t $_\tau$**
Gene 1	$GST_{1,1}$	$GST_{1,2}$	…	…	$GST_{1,\tau}$
Gene 2	$GST_{2,1}$	$GST_{2,2}$	…	…	$GST_{2,\tau}$
…	…	…	…	…	…
…	…	…	…	…	…
Gene n	$GST_{n,1}$	$GST_{n,2}$	…	…	$GST_{n,\tau}$

Table 4.3 *GST* score matrix.

We then use the GST scores to rank the genes as follows. Suppose table 4.3 is the *GST* score matrix.

	group$_1$	**group$_2$**	**…**	**…**	**group $_\tau$**
	t$_1$	**t$_2$**	**…**	**…**	**t $_\tau$**
Gene 1	$Rank_{1,1}$	$Rank_{1,2}$	…	…	$Rank_{1,\tau}$
Gene 2	$Rank_{2,1}$	$Rank_{2,2}$	…	…	$Rank_{2,\tau}$
…	…	…	…	…	…
…	…	…	…	…	…
Gene n	$Rank_{n,1}$	$Rank_{n,2}$	…	…	$Rank_{n,\tau}$

Table 4.4: *GST* ranking matrix.

	Control sample 1	Control sample 2	Exposure sample 1	Exposure sample 2
Gene 1	*0.95, 1.1, 1*	*1, 1, 0.9*	*1, 1.2, 1*	*1.05, 1, 1*
Gene 2	*1.1, 1, 0.9*	*1, 1.2, 1*	*1.3, 1.4, 1.3*	*1.3, 1.4, 1.4*
Gene 3	*1, 1.5, 1.9*	*0.9, 1.6, 2*	*1.6, 1.4, 1.4*	*1.5, 1.5, 2*
Gene 4	*0.9, 1, 1.1*	*0.7, 1, 1.2*	*1, 1.6, 2*	*1, 1.4, 2.1*
Gene 5	*1.2, 1, 0.9*	*1, 1, 1.1*	*2.3, 1, 1*	*2.4, 0.9, 1*
Gene 6	*1, 1, 1.3*	*1, 1, 1*	*1.1, 1.9, 1*	*1, 2.1, 1*
Gene 7	*1, 0.9, 1*	*1, 0.8, 1.05*	*1.1, 1, 2.5*	*1, 1.1, 2.3*

Table 4.5: An example *MTSO* dataset.

Data are in the order of (t_1, t_2, t_3)

	t$_1$	**t$_2$**	**t $_3$**
Gene 1	*1*	*0.333333*	*1*
Gene 2	*5*	*3*	*4*
Gene 3	*6*	*1*	*0.714286*
Gene 4	*2*	*5*	*12.33333*
Gene 5	*8.333333*	*1*	*0*
Gene 6	*1*	*18*	*1*
Gene 7	*1*	*3*	*14*

Table 4:6 *GST* score matrix using example data.

To illustrate, consider table 4.5 which consists of four *MTSO* matrices. This table will also be used as a running example for our algorithms when possible. In this example data set, two *MTSO* matrices belong to the "Control" class (also called negative class), and the other two matrices belong to the "Exposure" class (also called positive class). The associated *GST* and ranking matrices are also shown in table 4.6 and 4.7.

	t_1	t_2	t_3
Gene 1	*7*	*7*	*5*
Gene 2	*3*	*3*	*3*
Gene 3	*2*	*6*	*6*
Gene 4	*4*	*2*	*2*
Gene 5	*1*	*6*	*7*
Gene 6	*7*	*1*	*5*
Gene 7	*7*	*4*	*1*

Table 4.7: *GST* ranking matrix using example data.

When the time factor is not present, the rank of a feature is usually directly correlated with the feature's importance for classification. Our concern is: will the introduction of the time factor into the ranking process affect the ranking results as well as the classification results? If the answer is yes, why?

- To answer these questions, let us first have a look at our example. Figure 4.1 is drawn to examine the rank changes for every gene in the example. This chart shows that the rank for a gene depends on its time factor and the rank changes from time to time.
- We also examine the actual data set to see the impact of adopting different ranking systems into a given classifier. We will explain this in detail in the following section. The experiments show that using different ranking systems to the same test leads to different classification results.

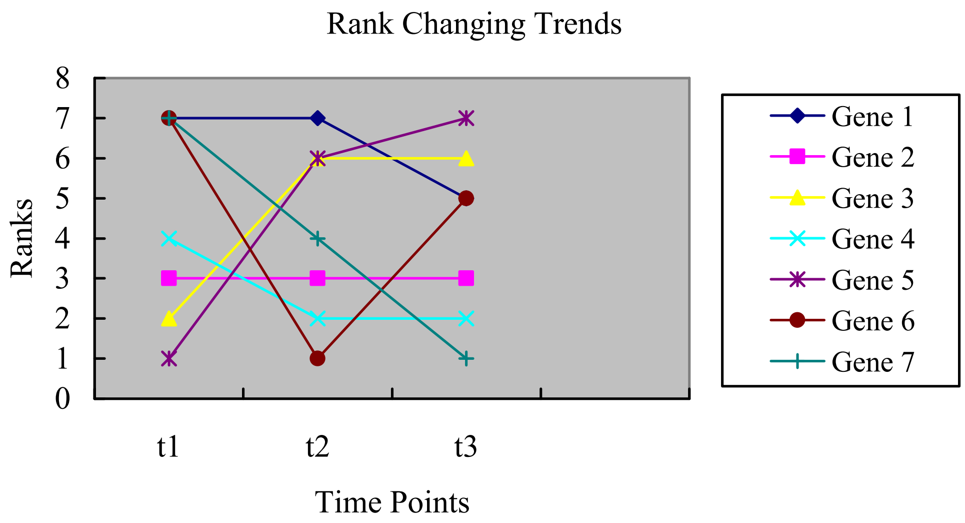

Figure 4.1: Rank changing trends of the example data.

According to the two tests we performed above, we can easily draw the conclusion that using a gene ranking result without considering its time factor is neither accurate nor scientific. For a given gene, at some time points the rank can be very high, which means this gene could be an informative gene that is highly correlated with its class. However, at some time points the rank of the same gene could be really low which means this gene is not closely correlated with the class label. Therefore, our next topic is how to accommodate the time factor in our new classification system.

4.2.2.2. Classification

In this section, we will focus on how to utilize the previous ranking results into classification. Before we illustrate our classification algorithm, let us define some prototypical *MTSO*s for future discussion convenience.

Suppose we are given a *MTSO* data set of two classes, *P* (Positive) and *N* (Negative). Let P_i denote the *ith* matrix of the *P* class, and similarly N_i for the *N* class. Suppose the *P* class has n_P *MSTO*s and the *N* class has n_N *MSTO*s. For each gene *g* and time point *t* let

$$Average_P[g,t] = \sum_{i=1}^{n_P} \frac{P_i[g,t]}{n_P},$$

$$Average_N[g,t] = \sum_{i=1}^{n_N} \frac{N_i[g,t]}{n_N}.$$

	t1	t2	t3
Gene 1	0.975	1.05	0.95
Gene 2	1.05	1.1	0.95
Gene 3	0.95	1.55	1.95
Gene 4	0.8	1	1.15
Gene 5	1.1	1	1
Gene 6	1	1	1.15
Gene 7	1	0.85	1.025

a. Example $Average_N$ *MTSO* data

	t1	t2	t3
Gene 1	1.025	1.1	1
Gene 2	1.3	1.4	1.35
Gene 3	1.55	1.45	1.7
Gene 4	1	1.5	2.05
Gene 5	2.35	0.95	1
Gene 6	1.05	2	1
Gene 7	1.05	1.05	2.4

b. Example $Average_P$ *MTSO* data

Table 4.8: Example average *MTSO* data.

The table 4.8.a is the average *MTSO* for the *P* class, whereas the table 4.8.b is the average *MTSO* for the *N* class. Observe that our method does not require that n_P be equal to n_N.

Similarly, we define a "*standard deviation MTSO*" for each class. For each gene *g* and time point *t* let

$$Variance_P[g,t] = \sum_{i=1}^{n_P} \left(\frac{(P_i[g,t] - Average_P[g,t])^2}{n_P} \right),$$

$$Variance_N[g,t] = \sum_{i=1}^{n_N} \left(\frac{(N_i[g,t] - Average_N[g,t])^2}{n_N} \right).$$

Table 4.9.a is the variance *MTSO* for the *P* class, whereas table 4.9.b is the variance *MTSO* for the *N* class. According to these two formulas, we can easily derive "*standard deviation MTSOs*": $Deviation_P[g, t]$ and $Deviation_N[g, t]$. Example data is shown in table 4.10.

	t1	t2	t3
Gene 1	0.000625	0.0025	0.0025
Gene 2	0.0025	0.01	0.0025
Gene 3	0.0025	0.0025	0.0025
Gene 4	0.01	0	0.0025
Gene 5	0.01	0	0.01
Gene 6	0	0	0.0225
Gene 7	0	0.0025	0.000625

a. Example $Variance_N$ *MTSO* data

	t1	t2	t3
Gene 1	0.000625	0.01	0
Gene 2	0	0	0.0025
Gene 3	0.0025	0.0025	0.09
Gene 4	0	0.01	0.0025
Gene 5	0.0025	0.0025	0
Gene 6	0.0025	0.01	0
Gene 7	0.0025	0.0025	0.01

b. Example $Variance_P$ *MTSO* data

Table 4.9: Example variance *MTSO* data.

$$Deviation_P[g,t] = \sqrt{Variance_P[g,t]},$$

$$Deviation_N[g,t] = \sqrt{Variance_N[g,t]}.$$

	t1	t2	t3
Gene 1	0.025	0.05	0.05
Gene 2	0.05	0.1	0.05
Gene 3	0.05	0.05	0.05
Gene 4	0.1	0	0.05
Gene 5	0.1	0	0.1
Gene 6	0	0	0.15
Gene 7	0	0.05	0.025

a. Example $Deviation_N$ *MTSO* data

	t1	t2	t3
Gene 1	0.025	0.1	0
Gene 2	0	0	0.05
Gene 3	0.05	0.05	0.3
Gene 4	0	0.1	0.05
Gene 5	0.05	0.05	0
Gene 6	0.05	0.1	0
Gene 7	0.05	0.05	0.1

b. Example $Deviation_P$ *MTSO* data

Table 4.10: Example deviation *MTSO* data.

Given the *GST* ranking results, we now consider how to use the top genes to build a classifier. It should be noted that the classifier deals with one time point only.

For a given time point *t*, let $P(t) = \{P_1(t), P_2(t), \ldots , P_\mu(t)\}$ be a set of positive (e.g.

exposed) microarray data, and let $N(t) = \{N_1(t), N_2(t), ..., N_\upsilon(t)\}$ be a set of negative (e.g. controlled) microarray data. The training input is of form $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_m, Y_m)\}$ for some unknown function $Y = f(X)$. A microarray data, X_j, either from set $P(t)$ or $N(t)$, is a vector of the form $< X_j(1), X_j(2), ..., X_j(G) >$. In this vector, $X_j(i)$ represents the expression value of *ith* gene, where $0< i <G$ and G is the total number of available genes. The domain of Y is normally a discrete set of classes, such as *{controlled, exposed}* in our case. After the training procedure, classifiers can be made so that when given a test microarray data, X, a Y value can be predicted as either controlled or exposed.

The detailed classification training method is described below:

Let g^{GTS} be the top gene under the *GTS* ranking system of time point *t*. Let X be a given unknown class sample microarray to be classified. We define three scores as follows:

$$Score_N(X) = \left|X(g^{GTS}) - Average_N(g^{GTS})\right|,$$

$$Score_P(X) = \left|X(g^{GTS}) - Average_P(g^{GTS})\right|.$$

$$Score(X) = \log_2 \frac{Score_P(X)}{Score_N(X)}.$$

To avoid division by zero situation,

$$Score(X) = \log_2 \frac{Score_P(X)}{Score_N(X) + \varepsilon}$$

is used. Here, ε is a very small positive number (e.g. *0.000001*). Also, we use function log_2 because of its ability to treat numbers and their reciprocals symmetrically so that we can view the results in a better perspective.

From the results of training data, we use an entropy based binning method to find the best cut-off threshold to decide the class of *X*:

- If $Score(X) > 0$, then *X* is classified as a member of *P* group;
- If $Score(X) < 0,$ then *X* is classified as a member of *N* group;
- If $Score(X) = 0$, then *X* is unclassified.

Here, zero is used as a threshold to separate all the cases. The rationale for the choice of the threshold is stated as follows: if *X* belongs to the *N* class, then $Score_N(X)$ tends to be small and $Score_P(X)$ tends to be big. As a result $Score(X)$ should be a value much bigger than *0*. (In our experiments, $Score(X)$ tends to be larger than *1*.) On the other hand, if *X* belongs to the *P* class, $Score(X)$ should be much smaller than *0*. (In our experiments it tends to be smaller than *-1*.)

In order to make the classification results more persuasive even though the number of unclassifiable cases will increase we will assign a range, [-1, 1], instead of using a number as the threshold.

- If $Score(X) >= 1$, then *X* is classified as a member of *P* group;
- If $Score(X) <= -1,$ then *X* is classified as a member of *N* group;
- Otherwise, *X* is unclassified.

We must emphasize that because we have data from n>1 different time points, we have *3n* different GTS ranking results. Now, we limit our classifications inside the same ranking system. In other words, if the top genes are chosen from the GTS ranking results that are associated to one time point, all the microarray data that are going to be used in the classification are also sampled at that time point.

4.2.3. Experiments

Let us briefly introduce the dataset we are going to use throughout our research. The dataset we used in our experiments was provided by Professor James McDougal in a scientific toxicology study field. As introduced in [60], the jet fuel jet propulsion fuel 8 (JP-8) has been shown to cause an inflammatory response in the skin, which is characterized histologically by erythema, edema, and hyperplasia. There are 9 *MTSO* data in this dataset: 5 of them belong to the controlled group and 4 of them were exposed to *JP-8*. Each time series has 3 time points: 1 hour, 4 hours, and 8 hours. This data includes 8799 genes. In our research we ignore the biological meanings of each gene and call each gene using their row number from the original data as well as the class labels assigned by biological researchers such as: A, M, and P. Research has also been done by Dr. McDougal by utilizing biological knowledge and some results are published in [60].

Table 4.11 and 4.12 show a small *GST* ranking result set. As you can see, gene 7070 ranks as the top 1 gene at time point 1. However, it ranks low at all other time points. Even though gene 2510 and gene 8468 seem better, neither of them always ranks in the top among three different ranking systems like gene 5519 does. This proves the point we mentioned earlier: the ranks of genes change greatly from time to time. Some research has been done to identify genes' responsive patterns such as immediate response, late response, intermediate response, and constant response. We will study this phenomenon in detail in the following sections.

Rank	Time point 1 ranking	Time point 2 ranking	Time point 3 ranking
1	7070	2510	8468
2	8156	5014	6895
3	7102	6895	5490
4	7447	2182	7078
5	6876	2944	1458
6	6899	3533	7204
7	2836	8126	7635
8	7071	4732	5671
9	2837	5650	8795
10	5898	7297	3967

Table 4.11: Top 10 genes at individual time point using *GST* method

Gene name	Time point 1 ranking	Time point 2 ranking	Time point 3 ranking
7070	1	8116	7380
2510	5696	1	2897
8468	1817	2553	1
5519	138	109	477

Table 4.12: Ranks of top 1 genes using *GST* method

In the next experiment genes from the *GST* ranking system are chosen for classifiers and results are shown in figure 4.2. In figure 4.2 we designed a chart to compare the overall performance of these genes selected from different ranking systems. In this chart there are three curves and each curve corresponds to classification using a ranking system: TP1, TP2, and TP3. The first 2000 genes are chosen from each ranking system and they are divided into 40 consecutive groups each with a size of 50. In each group genes are used in the classification individually and we count a gene as a satisfactory one only if all the classification results are correct. Otherwise it is unsatisfactory.

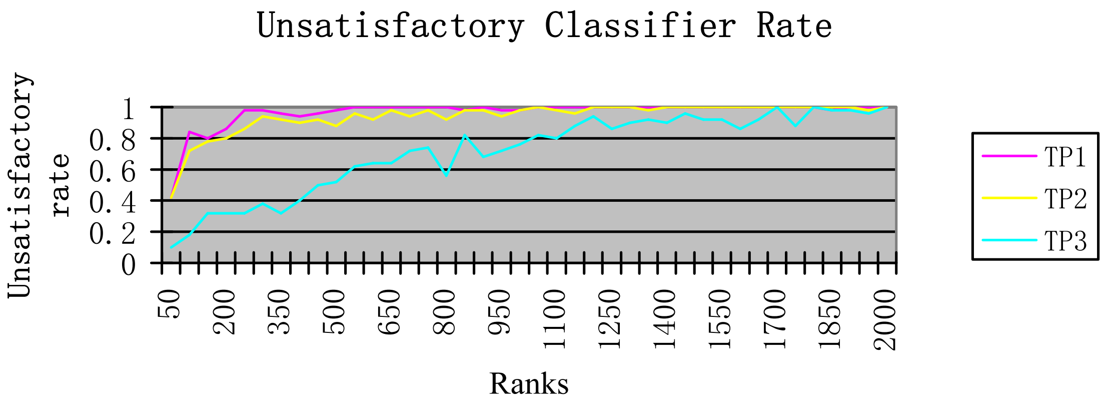

Figure 4.2: Ranking system comparison in microarray classification

According to this experiment, we can draw some conclusions:

- Each individual ranking system works as it is expected to. The unsatisfactory rates rise when the ranks get low. In other words, if one gene ranks higher than another gene the correctness rate is very likely to be higher if they are both used in the classifiers even though small fluctuations exist.
- The order of performance is: TP1 < TP2 << TP3. It is a very reasonable phenomenon because of the special character of living tissues: TP3 data are collected at 8 hours after exposure and the gene expression values have reached their relatively stable state if they have one. As a result, the genes that rank high in TP3 ranking system are relatively more closely correlated with the class label than top genes in other ranking system.
- But, the fact is, accuracy is not the sole concern in our research. For example, in order to better help a patient, doctors need to predict the cancer in its early stage. TP1 classification is just as important as any other time points even though its

performance is not as good as TP3 classification.

- These experiments are done according to their own input data time domain. In other words, these chosen genes are used to classify microarray data from the same time point. The performance is pretty good when considering the fact that even one single undetermined case will be considered as an unsatisfactory result.

The previous experiment shows that classification is working properly if the ranking information and test cases are from the same time point. Microarray data from different time points can also be selected to repeat the above experiment. In the following experiment, we try to conduct an experiment to see how classification performance suffers when the test data's time value is different from the training data's.

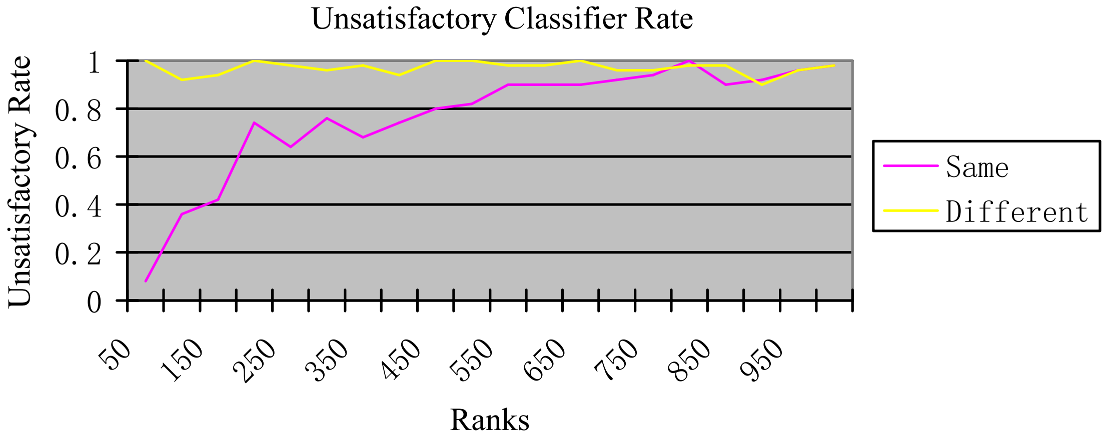

Figure 4.3: Ranking system comparison across time domain

Figure 4.3 shows the experiment results. In this figure, the curve with label "same" means that the features selected for the classifiers are selected from the training data for the same time point as that of the test microarray data. The curve with the "different" label means that the features selected for the classifiers are selected from the training data for some different time point from that of the test microarray data. We only considered classification situations where features are selected from time point 1 and test data are all associated with time point 2. Clearly, the results are totally unacceptable in the second case. The performance is extremely poor and the results show that even using the best feature from one time point to classify test data from another time point is inappropriate.

Hence, the experiments proved that rankings are not time independent. Given one time point and its corresponding ranking results, the genes selected for the classifiers can deliver good results if the test microarray data is chosen from the same time point. On the other hand, if test microarray data is chosen from different time points, the classification results become a disaster.

5. ET TIME SERIES RANKING AND MTSO CLASSIFICATION

Our focus in this chapter is to develop a classification approach that is applicable to *MTSO* data. In this method, the whole time series for each gene is treated as a feature. We will rank these features first and then classify the test *MTSO* data by using the top genes from the ranking results.

5.1. ET Feature Selection Using ET

5.1.1. Gene Ranking Method Motivation

Ideally, we want to find a feature which is discriminative at all time points. But, the fact is, often such a feature does not exist. Let us go back to the example in table 4.7 from the previous section. Genes 5, 6, and 7 rank at the top at time points 1, 2, and 3 respectively, and no gene ranks as the top 1 all the time, which means there is no one-suit-all solution. When such a situation arises, a method that ranks the whole time series is in need. So, we have to adjust our feature selection strategy by defining a ranking algorithm on the genes in the format of *MTSO*. Roughly speaking, this ranking system considers the whole time series instead of values at any single time point, and it favors genes that have large value differences among different classes and have small variances among values within each class at the same time point. In other words, the method developed here is to rank individual genes based on each gene’s correlation with its class.

5.1.2. Approach and Application

In this part, we have two sections. One is about the actual gene ranking on time series data and the other is about a pre-processing step that can be used to enhance the performance of ranking. We will talk about them in the reverse order because in order to recognize the necessity of the pre-processing, which is a screen procedure, we need to understand the actual ranking first.

5.1.2.1.ET Ranking

This *ET* gene ranking method and *GS/GST* method are somehow similar: they all take average values and deviations as part of the consideration. The main difference lies in whether they consider just one time point or all time points:

- The *GS/GST* methods are only applicable to single time point which means they can only analyze data from different time points separately.
- Our new method, *ET* Ranking, treats a whole time series as one data object which gives a method to compare two time series.

We will use *ET* as a ranking measurement over gene g to evaluate the correlations between the time series of gene g and the class label. It is defined as:

$$ET(g) = \sum_{i=1}^{\tau} \frac{|\, Average_P(g,t_i) - Average_N(g,t_i)\,|}{Deviation_P(g,t_i) + Deviation_N(g,t_i)},$$

where τ is the total number of available time points in the given time series.

	ET method		**GST method**		
	Score	**Rank**	**t_1**	**t_2**	**t_3**
Gene 1	*2.333333*	*7*	7	7	5
Gene 2	*12*	*4*	3	3	3
Gene 3	*7.714286*	*6*	2	6	6
Gene 4	*16*	*1*	4	2	2
Gene 5	*9.333333*	*5*	1	6	7
Gene 6	*12*	*4*	7	1	5
Gene 7	*14*	*2*	7	4	1

Table 5.1: Example *ET* ranking
(comparing with *GST* rankings)

In order to find appropriate informative genes for building classifiers for *MTSO* data in an accurate way, finding the most discriminating genes is necessary. Intuitively speaking, genes with larger *ET* values could be better candidates. These genes have large value differences between the two classes and they have relatively smaller variances within each class. As a result, we rank genes according to their *ET* values and the genes that rank on the top are more likely to be chosen in the classifier. Table 5.1 shows the *ET* ranking results for the example data.

But, experiments in classification show that *ET* rankings do not always help researchers to pick up the best candidates, which means genes with higher ranks are not necessarily better than the genes with lower ranks. We studied the data in further details and find out that the *ET* ranking system is not the sole standard, deviation plays an important role as well. So, we designed a screen procedure based on deviation value.

5.1.2.2.Filtering Using Deviation

This step is about screening available gene time series before actual ranking. In other words, this is a pre-processing step.

Genes are expressed differently from time to time and from group to group. In the ET ranking formula listed in the previous section, ET value is expressed as the sum of GST value from all the time points. But, we should notice that two genes may have identical GST values.

Here is an example. We have two genes, $gene_1$ and $gene_2$, and we also have their corresponding *GST* values, $GST_1 = \frac{5-4}{2+2}$ and $GST_2 = \frac{5-4}{0.2+3.8}$. For the above two genes, although the two average values of the two groups are the same, the sum of the two deviation values are the same, and the *GST* values are the same, we prefer the first gene because the deviation values of the second gene vary too much between the two groups (the deviation in the second group is 19 times the deviation in the first group). This preference is based on the following observation: bigger deviation means less accuracy in classification.

So, we decide to develop a screening procedure to find out the genes more like the first gene in the previous example. The intuition here is to find those genes that are consistently expressed at every single time point throughout the time series, either in control group or in exposure group.

We now consider how to find genes having the property just discussed. Several steps are needed for each gene *g*. Suppose we have 2τ deviation values: $Deviation_P[g,1]$, $Deviation_P[g,2]$, …, $Deviation_P[g,\tau]$ from the exposure group and $Deviation_N[g,1]$, $Deviation_N[g,2]$, …, $Deviation_N[g,\tau]$ from the control group.

1. Find out the minimum value, *MinDeviation*$_g$, from the deviation value set for gene *g*;
2. Find out the maximum value, *MaxDeviation*$_g$, from the deviation value set for gene *g*;
3. Let $DevRatio_g = \dfrac{MaxDeviation_g}{MinDeviation_g}$;
4. Set up a threshold value and eliminate all the genes whose $DevRatio_g$ is bigger than this threshold value.
5. Only the remaining genes will be considered in actual ranking algorithm illustrated in the previous section.

The above method can be further refined by considering both deviations and average values. The reason why we want to include average values into consideration is simple: larger values tend to have larger variances.

In the new method, we first define a *DAR* (Deviation-Average-Ratio) value for each gene *g* at every time point *t* in a group.

$$DAR[g,t] = \frac{Devation[g,t]}{Average[g,t]}$$

As a result, we still have 2τ *DAR* values for a given gene *g*: $DAR_P[g,1]$, $DAR_P[g,2]$, …,

$DAR_P[g,\tau]$ from exposure group and $DAR_N[g,1]$, $DAR_N[g,2]$, …, $DAR_N[g,\tau]$ from control group. Several steps are to be followed in the screen procedure:

1. Find out the minimum value, *MinDAR*$_{\mathrm{g},}$, from the *DAR* value set for gene *g*;
2. Find out the maximum value, *MaxDAR*$_{\mathrm{g}}$, from the *DAR* value set for gene *g*;
3. $DarRatio_g = \dfrac{MaxDAR_g}{MinDAR_g}$;
4. Set up a threshold value and eliminate all the genes whose $DarRatio_g$ is bigger than this threshold value.
5. Only the remaining genes will be considered in actual ranking algorithm illustrated in the previous section.

In practice, we need to avoid the situations such as minimum deviations being zero even though it is not normal when the size of data set is considerably big enough.

Time Series Rank	Gene	DarRatio	DevRatio
1	6895	10.92	2.56
2	8600	10.67	1.99
3	2324	13.62	2.43
4	1511	5.55	2.06
5	4453	14.59	2.39
6	6717	7.82	1.84
7	7078	11.53	2.12
8	3533	12.43	3.92
9	1393	10.14	2.65
10	223	7.97	2.1

Table 5.2: Top 10 *ET* ranking results and their *DarRatio* and *DevRatio* values

5.1.3. Experiments

We applied our ET ranking algorithm to our microarray time series dataset and some ranking results are shown in table 5.2.

Rank	Gene	DarRatio
1	4684	1.17
2	2858	1.18
3	6119	1.22
4	8745	1.23
5	5352	1.25
…	…	…
8795	957	45
8796	6973	52.31
8797	955	53.07
8798	7071	61.05
8799	944	72.4

Table 5.3: Top 5 and bottom 5 genes according to *DarRatio* values

Rank	Gene	DevRatio
1	1303	1.09
2	215	1.12
3	4312	1.15
4	6012	1.15
5	8768	1.15
…	…	…
8795	946	11.67
8796	957	13.47
8797	955	14.30
8798	2131	14.51
8799	944	19.85

Table 5.4: Top 5 and bottom 5 genes according to *DevRatio* values

Table 5.2 shows how genes are ranked in *ET* algorithm together with their corresponding *DarRatio* and *DevRatio* values. Combining with table 5.3 And 5.4, they give the readers some sense on what kind of *DarRatio* and *DevRatio* values are acceptable.

The classification experiments (will be studied in the next section) show that the *DarRatio* approach is not as persuasive as the *DevRatio* approach. The reason is obvious. For two genes that have the same deviation values all the time between two groups, the one that has bigger average expression value changes definitely produces bigger *DarRatio* values. As a conclusion, *DevRatio* is a more reliable screen approach in our case and *DarRatio* is more suitable for original data that has not been normalized yet.

5.2. Classification on MTSO Data Using ET Ranking

In the previous section, we provided a whole-time-series based ranking function, namely *ET* Ranking. In this part, we will consider approaches to *MTSO* data classification by utilizing the *ET* Ranking system, evaluate the performance in classification, and demonstrate the value of our *ET* ranking system. After that, we will use different approaches in building the classifiers: single feature or feature committee. The problems we try to answer are:

- How to classify *MTSO* data?
- Is *ET* Ranking actually better than *GST* ranking?
- How to improve the classification results when it is not accurate?
- Is our classification approach better than SVM?

There is one thing we want to point out: because our data set is relatively small and the feature set is relatively huge, it may be a good idea to obtain a large data set to reconfirm the results of experiments reported here.

5.2.1. Classification Method

The results from the *ET* ranking system can be used in different ways for different classification scenarios. We will consider two such ways and possible extensions:

- Classification using single feature
- Classification using feature committee
- Possible extension of ET ranking

Given the *ET*-Ranking system, we now consider how to use the top genes to build a classifier for *MTSO* data. More specifically, we consider the following supervised learning problem for *MTSO* data:

Let $P = \{P_1, P_2, \ldots,, P_\mu\}$ be a set of positive (e.g. exposed) *MTSOs*, and $N = \{N_1, N_2, \ldots, N_\upsilon\}$ be a set of negative (e.g. control) *MTSO*s. The training input is of form $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_m, Y_m)\}$ for some unknown function $Y = f(X)$. An *MTSO* data, X_j, is a matrix where $0< j <n+1$, which can be expressed as a vector of the form $< X_j(t_1), X_j(t_2), \ldots, X_j(t_\tau) >$. In this vector, $X_j(t_i)$ represents *ith* column of matrix X_j and it is a microarray collected at time t_i, where $0< i < \tau +1$. The domain of *Y* is normally a discrete set of classes, such as *{control, exposed}* in our case. After the training procedure, classifiers are built. Given a test *MTSO* data, *X*, and a *Y* value can be predicted by such classifiers as either control or exposed.

5.2.1.1.Classification Using Single Feature

5.2.1.1.1. Classification Approach

The detailed classification training method is described below:

Let *g* be a gene, and let *X* be a given sample *MTSO* to be classified. We define three scores as follows:

$$Score_N(X) = \sum_{i=1}^{\tau} dist(X(g\ \ ,t_i) - Average_N(g\ \ ,t_i)),$$

$$Score_P(X) = \sum_{i=1}^{\tau} dist(X(g\ \ ,t_i) - Average_P(g\ \ ,t_i)).$$

$$Score(x) = \log_2 \frac{Score_P(X)}{Score_N(X)}.$$

Similar as before, to avoid division by zero, the following formula can be used instead of the last one given above.

$$Score(x) = \log_2 \frac{Score_P(X)}{Score_N(X) + \varepsilon}$$

Here, ε is a very small positive number (e.g. *0.000001*), and *dist* is a distance function that measures the difference between two vectors. We have two choices of the distance function:

1. Manhattan distance: The distance between two points measured along axes at right angles. In a plane with P_1 at *(x_1, y_1)* and P_2 at *(x_2, y_2)*, it is $|x_1 - x_2| + |y_1 - y_2|$.
2. Euclidean distance: The straight line distance between two points. In a plane with P_1 at *(x_1, y_1)* and P_2 at *(x_2, y_2)*, it is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Theoretically speaking, using different distance functions may lead to different score values. However, our experiments show that they don't make much difference. So we use Manhattan distance throughout our research.

From the classification results on the training data, we use entropy based binning method to find the best cut-off threshold to decide the class of *X*. In particular, -1/1 can be used as the thresholds.

- If *Score(X) >= 1*, then *X* is classified as a member of the *N* group;
- If *Score(X) <= -1,* then *X* is classified as a member of the *P* group;
- Otherwise, X is unclassified.

Other values can also be used as thresholds, such as -0.5/0.5.

5.2.1.1.2. Experiments

We plan to compare the performance of different ranking systems for classification. To this end, we present table 5.5, which shows the ranks of some genes under the ET ranking. The GST ranks of these genes are also shown so that readers can have an idea on how ranking results from different ranking systems are related.

Gene name	Time point 1 ranking	Time point 2 ranking	Time point 3 ranking	Time series ranking
7070	1	8116	7380	28
2510	5696	1	2897	252
8468	1817	2553	1	12
6895	1665	3	2	1
7680	6469	73	7611	2225
8009	5198	7258	570	2058

Table 5.5: Ranking results comparison of *ET* and *GST*

Table 5.6 shows the classification performance of those genes. It should be noted that some of the genes are not ranked near the top. For example, gene 2510 was ranked at 252. However, this gene's relative rank is pretty high, in the top 2.8% (the number of available genes is a huge number.)

Gene name	Correct results	Undetermined	Incorrect results
7070	9	0	0
2510	8	1	0
8468	7	2	0
6895	9	0	0
7680	1	8	0
8009	6	2	1

Table 5.6: *MTSO* classification performance

Results from table 5.6 show that:

- Genes that rank on the top of ET ranking results normally give pretty good *MTSO* classification results.
- Genes that rank on the top of GST ranking results do not necessarily put them on top of ET ranking results, and do not necessarily mean they can give good *MTSO* classification results.

Here we will make a comparison of the four different ranking systems based on their performance in the *MTSO* classification: TP1, TP2, TP3, and ET. The purpose is to evaluate which ranking system lead to more accurate classification and what kind of results do we expect when using the wrong ranking system. In this way, we can see the value of ET ranking systems in *MTSO* classification. In this experiment, the first 500 genes are chosen from each ranking system and they are divided into 10 consecutive groups of size 50. In each group, genes are used in the classification individually and we count a gene as a satisfactory feature if all the classification results are correct. Otherwise, it is unsatisfactory. In this experiment, -0.5/0.5 is used as the threshold.

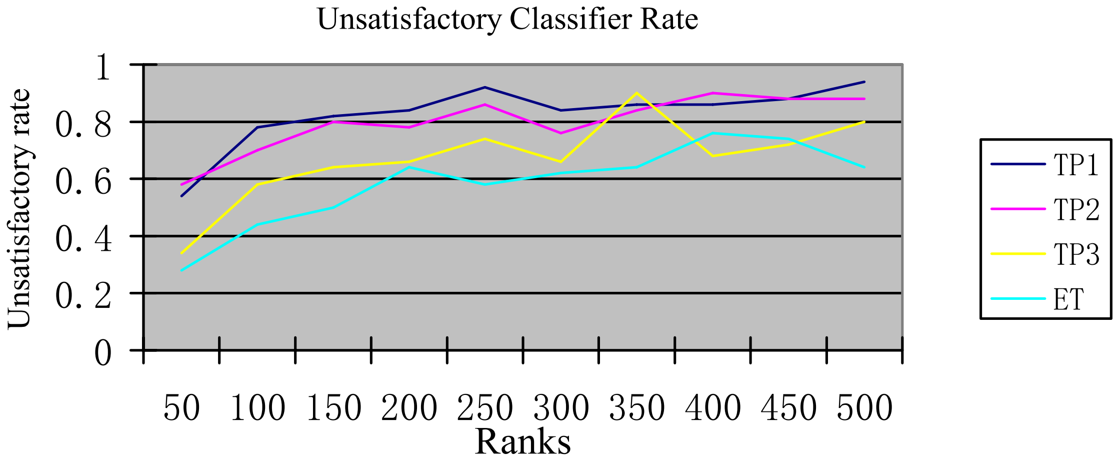

Figure 5.1: Ranking system comparison in *MTSO* classification

From figure 5.1 above, we can make certain conclusions:

- Same as before, each individual ranking system (TP1, TP2, and TP3) works reasonably well even though they were not designed for dealing with *MTSO* data. The unsatisfactory rate rises when the rank gets low. In other words, if one gene ranks higher than another gene, the correctness rate is very likely to be higher as well.
- The performance sequence is: TP1 < TP2 < TP3 < ET. This verifies our original intuition that our ET ranking system is better than other ranking systems in *MTSO* classification. It is interesting to note that, although the ET ranking system is a linear combination of the other three, it offers better performance than all the other three.
- We also notice that TP1 < TP2 < TP3, which is in the same order as figure4.?.

From the results shown above, we learned that even though the ET ranking system is not necessarily the best solution at any individual time point, it is the best solution so far for the whole time series.

In practical situations, choosing only one feature from the ET ranking system may not enable us to achieve satisfactory performance in *MTSO* classification. There are approaches that can improve the *MTSO* classification accuracy: we can either organize a decision committee or further improve the ranking formula. We will introduce these ideas in the following two sections.

5.2.1.2. Classification Using Feature Committee

Decision committee learning has been proven to be very effective in reducing classification error from learned classifiers, both practically and theoretically. The committee members are given the same classification task and their individual outputs are combined in a certain way to create the final conclusion. This combination of outputs is usually obtained by majority vote.

Here, we will briefly discuss how to choose these committee members and how to combine the results from these committee members in details.

5.2.1.2.1. Organizing Decision Committee

There are two options in choosing committee members: we can either choose the top *c* genes from the ET ranking system, or we can choose the top c/k genes from each of the ranking system. Here *c* stands for the number of committee members and *k* stands for the number of available ranking systems. In order to make our committee members more diversified, we may want to use the second approach, even though features from single time point ranking system may not be as superior as the features from ET ranking system. Here, by superior, we mean “leading to more accurate *MTSO* classification results.”

5.2.1.2.2. Voting

Here, we consider ways of combining the classification results that come from these committee members so that final conclusion can be made. Suppose in our feature committee, we have *c* genes: $g_1, g_2, ..., g_c$. These genes can be either chosen from different ranking systems or just chosen from ET ranking systems. According to our classification algorithm, when given test data, *X*, each committee member will give a score: $Score_1(X), Score_2(X),, ..., Score_c(X)$, and their corresponding classification results are: $Y_1(X), Y_2(X), \ldots, Y_c(X)$ where $Y_i(X) \in \{-1, 0, 1\}$. Here, -1 stands for negative, 0 stands for undetermined, 1 stands for positive.

We have two options:

- Equal Vote. We can use formula

  $$Y(X) = Y_1(X) + Y_2(X) + ... + Y_c(X)$$

  to finalize our classification result. In this formula, every vote from the members is given the same weight. Then we use $Y(X)$ to determine the class label of data

X:

- If $Y(X) < 0$, then *X* is classified as member of *N* group;
- If $Y(X) > 0$, then *X* is classified as member of *P* group;
- If $Y(X) = 0$, then *X* is unclassified.

- Weighted Vote. In the equal vote option, we treat these votes indiscriminately: members have equal vote towards the final conclusion. Since different ranking systems are associated with different classification accuracy, and since genes with different ranks in a given ranking are associated with different classification accuracy, the votes should not be treated the same in the counsel. We decide to arrange their contributions to the final score according to their ranks or scores.

$$Score(X) = \frac{Score_1(X) + Score_2(X) + ... + Score_c(X)}{c}$$

or

$$Score(X) = \frac{k_1 * Score_1(X) + k_2 * Score_2(X) + ... + k_c * Score_c(X)}{c}$$

$$k_i = 1 - \frac{ETRank(g_i)}{G}$$

where G is the total number of genes in the *MTSO*, k_i is the weight parameter of g_i in the committee, and $ETRank(g_i)$ is the ET rank of g_i. For the top gene in the ranking results, it will be given the highest weight, 1. For the bottom gene in the ranking results, it will be given the lowest weight, 0. This is a linearly decreasing weight function. Of course, different weight functions can be defined, such as exponential, or log. But, experiments show that these weight functions produce very similar performance.

5.2.1.2.3. Experiments

The following experiment is done to demonstrate the effectiveness of using committee in the classification. Equal voting is done in this experiment. 3000 committees are organized. Each committee has 4 members from 4 different ranking systems so that every ranking system has its representative in the committee. To be more specific, committee *i* consists of four features: *i*th feature from each ranking system, TP1, TP2, TP3, and ET.

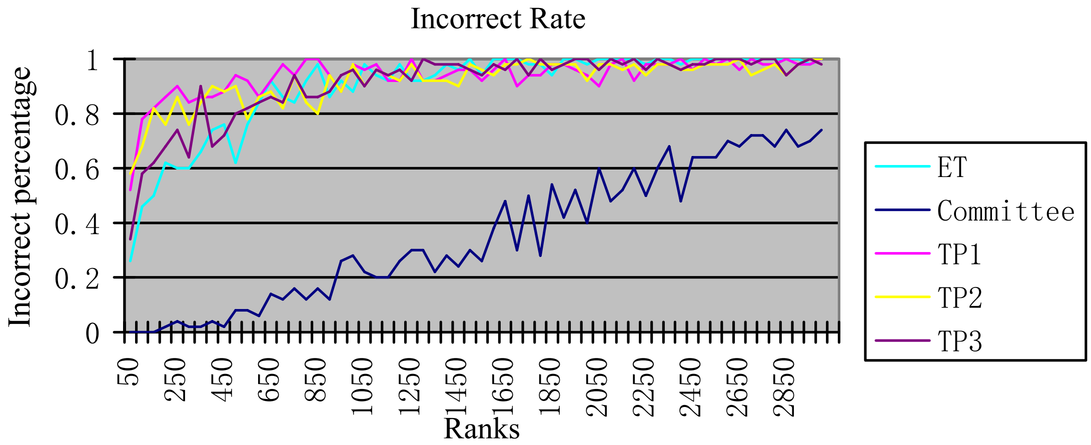

Figure 5.2: Ranking system comparison in *MTSO* classification (with committee)

Figure 5.2 shows that the classification performance is very stable and is significantly improved by using voting committees. We now give some more specific numbers, since the figure above is not clear enough. For the top 50 genes from each ranking system:

- 24 genes from TP1 ranking system give satisfactory results;
- 21 genes from TP2 ranking system give satisfactory results;
- 33 genes from TP3 ranking system give satisfactory results;

- 37 genes from ET ranking system give satisfactory results;
- But, when these genes are combined into 50 committees, all of these committees are quite accurate. In fact, the first 150 committees are 100 percent accurate.

5.2.1.3.Extension of ET Ranking

5.2.1.3.1. Approach

Theoretically, the classification performance can also be improved by revising the ET ranking formula even though the performance improvement is normally not as significant as the voting committee approach. One way to revise the ET ranking is to use the following ranking formula which gives different weights to different time points:

$$ET(g) = \sum_{i=1}^{\tau} k_i \frac{|\,Average_P(g,t_i) - Average_N(g,t_i)\,|}{Deviation_P(g,t_i) + Deviation_N(g,t_i)}$$

where k_i stands for the actual weight for time point i. The larger value k_i is, the more influential this time point is over the final ranking results.

Apparently, in the original formula, we assigned equal weights to every time point *(k_i =1)*. Here, we will provide more choices and analyze their potential meanings (they are also listed in figure 5.3):

1. Constant weight function (equal weight for each time point). In such a weight function, changes at different time points are treated equally.
2. Linearly decreasing weight function. In such a weight function, changes at the beginning of the time series are magnified and hence play a bigger role than changes

towards the end of the time series.

3. Linearly increasing weight function. In such a weight function, changes at the end of the time series are magnified and hence play a bigger role than changes at the beginning of the time series.

4. Quadratic function. In such a weight function, changes at the middle of the time series are either magnified or diminished and hence play a bigger or smaller role than changes at either the beginning or the end of the time series.

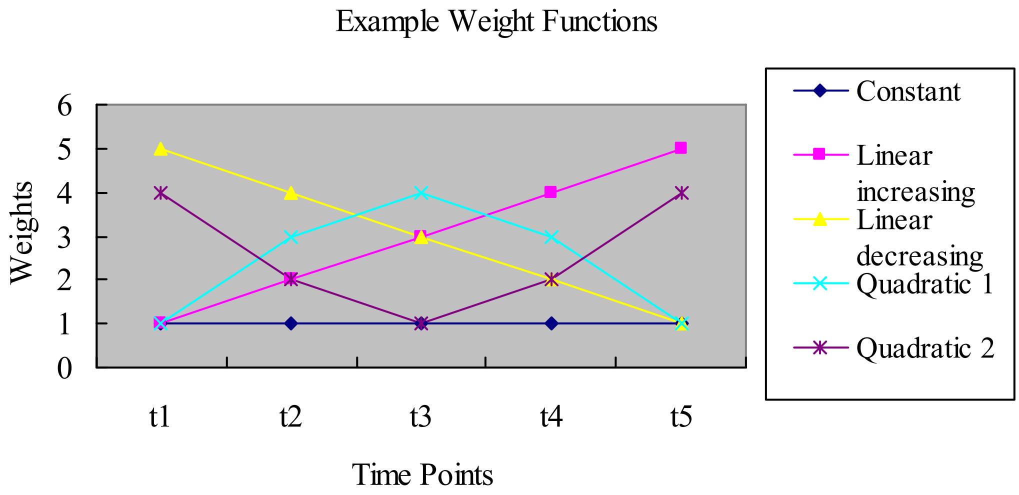

Figure 5.3: Weight functions

A universal weight function that fits all different requirements does not exist. In other words, the optimal weight function depends on the application. Choosing an appropriate weight function for a given application is important because data at different time points may play different roles and have different significance.

5.2.1.3.2. Experiments

In this experiment, different weights are given to different time points in ET ranking.

Then, these ranking results are applied in *MTSO* classifiers and the performance is compared.

Table 5.7 is used to show how the rank changes among three different weight functions: ET123 (linearly increasing), ET111 (constants), and ET321 (linearly decreasing).

Rank	ET123	ET111	ET321
1	6895	6895	7070
2	8600	8600	8156
3	7078	2324	8270
4	8468	1511	7102
5	5671	4453	5898
6	3533	6717	6895
7	1393	7078	2324
8	1458	3533	1511
9	4453	1393	6876
10	5490	223	7447

Table 5.7: Top 10 *ET* results using different weights

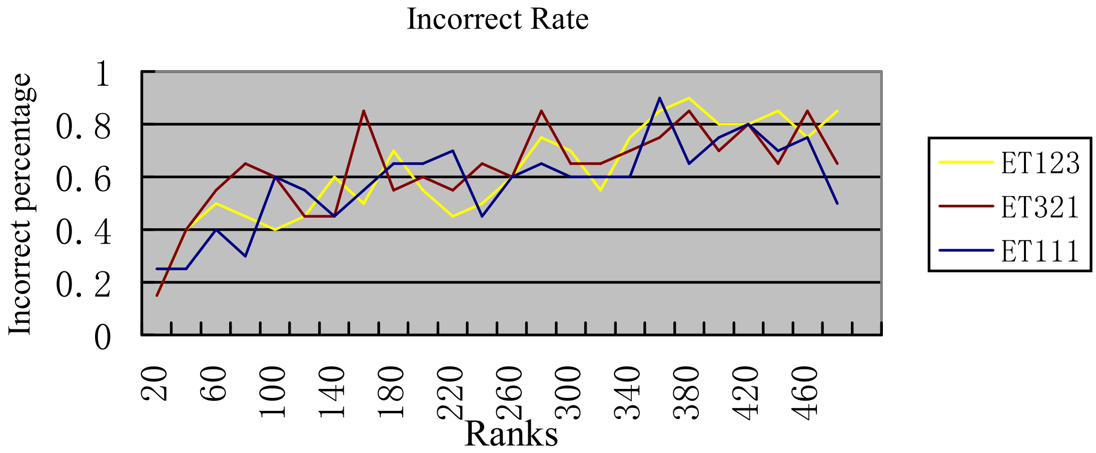

Figure 5.4: Performance comparison using different weights

From figure 5.4, we can see that these three different weight functions don't lead to any major difference in the classification results for our *MTSO* dataset.

5.2.2. Discussion

There are other classification approaches that can be easily extended to classify *MTSO* data, such as SVM. So, we conduct a performance comparison to answer questions such as which one is superior. The software we are using is called SVMdark provided by Martin Sewell. SVMlight is an implementation of Support Vector Machines (SVMs) in C and SVMdark is a Windows implementation of SVMs based on SVMlight.

SVM is a method to find a hyperplane to separate the training set in a high dimensional feature space. Importantly, it is proved that the data set we are using is linearly separable. SVM is a preferred method for classification in general, even though it has disadvantages such as too many parameters, not easy to use, slow.

Our experiments show that, after finding the correct parameter in training, the SVM method gives quite satisfactory results, just the same as our ET ranking approach.

The usage of SVM is limited because of various reasons such as its tedious training procedure and the slow speed. But, most of all, researchers are more interested in biologically interpretable results whenever possible, not just black-box classification results. So, statistical significant results from SVM are not good enough for us. Marker genes are genes whose expression values are biologically useful for determining the class of the samples. The identification of marker genes is important due to the following reasons:

1. Enable better classification performance.
2. Allow biologists to further study the interaction of relevant genes in achieving a certain biological performance.
3. Study the functional and sequential behavior of known marker genes in order to facilitate the functionality discovery of other genes.
4. Allow further study of relation of expression values of different genes with respect to the tumor class, similar expression pattern always results in cancer or the combination of suppression of certain genes and expression of certain genes are a better indication of tumor, etc.
5. Using fewer genes to save money in test-kit design and production. It can also be time efficient for circumstances such as the battle field.

As a result, our algorithm is superior to SVM in speed and the capability of identifying gene markers.

6. FRAGMENT CLASSIFICATION AND TIME RECOVERY

In this chapter, we introduce a new problem: how to classify partial *MTSO* data. To be specific, one *MTSO* data has several microarray data and each of them has a time value. The previous classification approaches are not applicable if the time value for the microarray data is missing. We call such a problem fragment classification. To solve this problem, we need to find time insensitive features and then use them to classify the partial *MTSO* data. We will also discuss how to predict the missing time value, which is a natural extension of the fragment classification.

6.1. Robust Feature Set

6.1.1. Motivation

In the previous sections, we discussed how to select the informative genes that could be used in regular classifiers for *MTSO* data and how to use these genes in *MTSO* classifiers. The *ET*-Ranking system gives us a new metric on ranking time series and extends the targets of gene ranking algorithm from single time point microarray data to the *MTSO*. Our experiments from the previous section show that the informative genes obtained by ranking time series work quite accurately and reliably for *MTSO* classification.

However, *MTSO* classification is not the sole usage of *ET*-Ranking system. *ET* Ranking results can be interpreted in different ways and can be effective under various situations. Moreover, changing or adding parameters can make it more applicable for certain applications. One such possible application is called fragment classification.

Fragment classification is a non-regular classification problem. The basic idea is how to classify a partial *MTSO* data. We are especially interested in the situation where the time value of some data is missing. To solve this problem, we need to find ways to spot time insensitive and robust features so that they can deliver reasonably accurate and reliable classification results in different situations. Here, robustness is defined as the capability of performing classification without failure under a wide range of condition.

Our approach involves the following steps. Based on time series data, we first group features that have consistency as their common characteristics and we call this group of features the robust feature sets. We will discuss fragment classification using these robust features in the next section.

Before giving the details of feature selection, let us use microarray time series data as an example to explain why it is possible to find robust features. In microarray time series data, different genes respond to certain circumstance changes in different manners.

- Some genes are early responsive, which means these genes are expressed significantly different from normal at the early stage after the exposure but the difference diminishes after a certain period. These genes are good feature candidates for fragment classification if the test data is also in the early stage after the exposure, but not the other way around.

- Some genes are late responsive, which means the reactions of these genes to the exposure have some latency. They may produce very good accuracy when the testing data is from patients who are in the late stage of exposure, but it could be disastrous when applied to other patients.
- Some genes react to these changes in a consistent manner. These genes are the key factors in fragment classification. We will focus on these genes' either rank based or value based consistency. Obviously, these genes are not as accurate or sensitive as the early responsive genes when used to classify early stage data and they are not as good as late responsive genes when dealing with late stage data either. Their advantage is the capability to deliver stable results.
- Some genes are totally irresponsive to these changes. These genes are not helpful in fragment classification research.
- Some genes respond to the exposure in a random way, or should we say, in an unpredictable way. These genes are not helpful for fragment classification either. We will study these genes in a later section.

6.1.2. Robust Gene Marker Set

6.1.2.1. Rank Based Robust Gene Marker Set

In this section, we will introduce methods that help us to identify rank based robust feature sets. We use the ranking results from previous ranking systems (*GST* ranking and

ET ranking) and their associated time values as the sole information source and rank features according to the consistency level of their ranks. The top features in the ranking results will be treated as rank based robust feature sets. The keys of this approach are:

- We can try to find the features (genes) that are consistently ranked at the top in all the available ranking systems.
- We can try to find a diversified set of features (genes) that are responsive to the toxic compound, which, collectively, provide highly reliable predictions at all times. Here, by diversify, we mean that we want all available ranking systems to have their own representatives in this set.

The first approach is definitely a solution even though it is not always feasible. If the first option does not generate any or enough good feature candidates, we can switch to the second option. In the second option, we don't need these representatives to be the best choices from each ranking system because we have to consider their consistency. In fact, we only need these features to be reasonably good choices for the classifiers at each ranking system. In the second option, a decision tree or voting committee is required.

Gene name	$Rank_{t1}$	$Rank_{t2}$	$Rank_{t3}$	$Rank_{t0}$
7070	1	8116	7380	29
2510	5696	1	2897	252
8468	1817	2553	1	12
6895	1665	3	2	1
5519	138	109	477	15

Table 6.1: Rank information of top genes from different rankings

So far, for a time series of τ time points, we have $\tau + 1$ different gene ranking results: one *GST* for each time point plus one for the time series (*ET* ranking). The *GST* rank of gene g_k at time point t_i is represented as $Rank_{ti}(g_k), i \in [1, \tau]$. For the sake of convenience, the *ET* rank of gene g_k is represented as $Rank_{t0}(g_k)$. Table 6.1 gives some ideas on how the ranks of certain genes vary from time point to time point.

We have two concerns in rank based robust gene selection algorithms: the best performance and the consistency. We need to select those genes that are ranked very high in one ranking system and are ranked consistently high in all other ranking systems. We design four metrics to screen the candidate gene, g_k:

1. The highest rank of the gene among all ranking systems. This item represents the best performance a gene can achieve. (To make things clear, by "highest" genes, we mean it has the smallest rank value. For example, 1 is the highest possible rank.)

   $$MinRank(g_k) = \min(Rank_{t_0}(g_k), Rank_{t_1}(g_k), ...Rank_{t_\tau}(g_k)).$$

2. The lowest rank of the gene among all ranking systems. This item represents the worst performance a gene can have.

   $$MaxRank(g_k) = \max(Rank_{t_0}(g_k), Rank_{t_1}(g_k), ...Rank_{t_\tau}(g_k)).$$

3. Rank consistency of the gene. It is defined as the maximal value of the differences among the ranks of the gene over different ranking systems. This item represents how consistently this gene ranks throughout the whole time series.

   $$DiffRank(g_k) = MaxRank(g_k) - MinRank(g_k)$$

4. We choose to use the following formula

   $$RBRobustness(g_k) = MinRank(g_k) \times DiffRank(g_k)$$

   to measure how robust a gene is throughout the whole time series. A gene that has small *MinRank* value and small *DiffRank* value will be more likely to have small *RBRobustness* value; the genes that have large *MinRank* values and large *DiffRank* values are more likely to have large *RBRobustness* values. We rank all the genes according to their *RBRobustness* values and only the genes on the top are chosen as the rank based robust gene markers sets.

Generally speaking, biological researchers are more willing to rely on genes that behave in a stable and consistent manner. Such genes allow biologists to make their experiments more predictable and to make their theory more persuasive and reliable. The robust gene markers tend to have such a quality.

6.1.2.2. Value Based Robust Gene Marker Set

In this part, we will introduce methods that help us to find value based robust gene marker set. We will focus on the expression values and rank genes according to the consistency level of their expression values. The top genes in the ranking results will be treated as value based robust gene marker set.

The key in finding the value based robust gene marker set is similar to *GST* ranking system: find the features (genes) that are consistently expressed at all available time points in each class and are expressed greatly differently between different classes. Compared to the algorithm in finding rank based robust gene marker set, the value based algorithm is relatively easy.

We will continue to use *SNR* formula to retrieve the *VBRobustness* value for each gene *g*,

$$VBRobustness(g) = \left| \frac{\mu_1(g) - \mu_2(g)}{\sigma_1(g) + \sigma_2(g)} \right| .$$

Here, $\mu_1(g), \mu_2(g)$ and $\sigma_1(g), \sigma_2(g)$ stand for the mean and standard deviation of normalized original expression values of gene *g* in class 1 and 2 respectively. The formula is the same as the one used in the *GS* method. However, in *GS/GST* method, there is no specific requirement that asks for equal number of training data at each time point. In *VBRobustness* ranking, it will be ideal to have every time point contributes equally to the final candidate list. As a result, we will try to have roughly the same number of training data at each time point. Otherwise, we may have to assign different confidence parameters to their corresponding candidates as a compromise.

After we calculate these *VBRobustness* values, we rank the genes by using these values. Only the genes on the top are chosen as the value based robust gene markers sets.

At this point, we want to briefly introduce an interesting phenomenon called Wildness, which we will study later. Both rank based and value based approaches may not be always feasible and may not be optimal either. The key idea of these two approaches is to find a feature *g* such that it is highly uniformly ranked/expressed inside each class and its values vary greatly between different classes. Problems emerge when such features don't exist, or when the fragment classification performance by using such features is not satisfactory. When this problem appears, we must look for alternative solutions. After observation, we found that there are genes that are highly uniformly expressed in the normal class. However, their values in the exposure class vary greatly and are unpredictable. We call this phenomenon "Wildness". By spotting these wild genes, we can still build satisfactory classifiers. We will discuss how to locate and use these genes in later sections.

6.1.3. Experiments and Discussion

In the first experiment, we rank all the genes according to their *RBRobustness* values and the following table shows the ranking results.

Robustness Ranking	Gene name	$Rank_{t1}$	$Rank_{t2}$	$Rank_{t3}$	$Rank_{t0}$
1	1511	93	120	129	3
2	8600	358	45	15	1
3	4453	200	366	19	4
4	2324	57	506	29	2
5	5519	138	109	477	14
6	1112	165	427	313	29
7	223	592	98	85	9
8	1900	292	86	539	31
9	4113	118	216	573	26
10	2223	444	553	64	22

Table 6.2: Example top genes from rank based robustness ranking

From table 6.2, we notice that the rank based robustness ranking results are relatively similar to the ET ranking results.

In the second experiment, we rank all the genes according to their *VBRobustness* values and table 6.3 shows the ranking results.

It is not a surprise to see that some genes, such as 1511, 8600 and 2324, appear on the top of both ranking systems. These genes are highly uniformly expressed in both classes, and consequently, they have high ranks in the value based robustness ranking system. Also, because they are highly uniformly expressed in both classes, they are highly uniformly expressed at every time point as well; so they have high ranks in the rank based robustness ranking system as well. There is no doubt that such genes are going to fit very well in the fragment classification that we are going to introduce in the next section.

Robustness Ranking	Gene name	*VBRobustness* value
1	1511	2.08071
2	8600	1.75879
3	6717	1.74582
4	1837	1.73577
5	1900	1.66149
6	2057	1.64648
7	4113	1.63108
8	2324	1.62978
9	7345	1.62976
10	4196	1.61431

Table 6.3: Example top genes from value based robustness ranking

6.2. Fragment Classification

In previous sections, we considered how to choose top features from different ranking systems for every time point and how to use these features in microarray data classification. (We should all understand that every microarray data has a time value. This time value is the elapsed time period since the possible exposure. We call it "time value" for the purpose of convenience. The time values we have been using are 2, 4, and 8 hours.) The approaches considered so far are based on the assumption that the time value of the test microarray data is known. This assumption allows us to choose the right features that correspond to the time value of the test data. However, there are cases where the time value of the test data is unknown and cases where the time value of the test data does not match that of any training data. This brings difficulties in performing regular classification. We call such a classification problem ***fragment classification***. In this section, we will define what fragment classification is, discuss why we rely on robust features, propose solutions to solve this problem, and finally evaluate the performance of our solutions.

6.2.1. Definition and Motivation

First, let us illustrate our idea using the example of hologram. A hologram is a three-dimensional image created with photographic projection by recording not just the intensity but also the phase information of light. When the hologram is illuminated by appropriate light, the entire three dimentional scene can be reconstructed. An interesting phenomenon of hologram is: if a laser light beam illuminates only a small part of the hologram, the entire image still appears, although it is now less refined and less detailed. This means that every portion of the hologram carries information about the entire image. It is interesting to compare this with normal photographs, where each portion of film contains only a corresponding portion of the whole image. Put simply, if we hold a fragment of a hologram, we are still able to make inferences on the entire original image. This is where the name of “fragment classification” comes from.

Now, we are going to bring the same concept into classification problems. In regular classification, in order to identify the class label, the test data and the training data need to have the same standard format, which means the loss of any part of the test data is detrimental. However, it is not unusual that only part of the original test data is available and we still want to know the class label of this partial data. Consequently, this problem cannot be handled by regular classification approaches. Here, we name this problem as fragment classification.

Fragment classification is a new classification challenge. The training procedure is approximately the same as the *MTSO* classification method. However, the test data is only a fragment of the *MTSO* matrix. We expect to be able to draw the same classification conclusion as if we were using the entire *MTSO* data. The size of the fragment can vary from a single column to several columns of the whole data matrix. Apparently, the smaller the fragment is, the more difficult the problem is. In our study, we use only one column, *X(t)*, as the testing input instead of the whole *MTSO*, *X*. Here, *t* is the time value of this array. The time variable can also take any values in a range even though they are not present in the training data. For example, the time values in the training microarray data set may be limited at 2, 4, 8 hour time points, but the time values of the test data can be 1, 3, 5, 7 etc hours.

The fragment classification approach can be used in various complicated applications. For example: to test the impact or existence of a toxic, microarray data is obtained from a living tissue under treatment. If we know the exact time when this tissue was treated or exposed, by using the earlier described microarray classification approach, we will be able to conclude if this tissue has been exposed to a toxic compound, and even classify which toxic has this tissue been exposed to. However, the situation is different if the time value of such a microarray is uncertain and this is quite likely to be true when detecting toxics in the battlefield. In other words, under circumstances that the possible exposure time is unknown or missing, it is hard to use previously discovered gene markers because of the time uncertainty.

After revisiting figure 4.3, it is obvious that classification performance suffers without using robust features when the test data's time value is different from the training data's. To produce more accurate classification results in fragment classification, finding the appropriate gene(s) for the fragment classifiers is crucial. In the next section, we will use the robust features to solve this problem.

6.2.2. Classification Method

The classification procedure is the same as the method we described in Chapter 5. However, experiments show that classification using single robust gene is generally not sufficient. A committee can be constructed to improve the accuracy requirement. In such a committee, each member has a vote and the final classification decision is made according to the majority rule.

Because of the natural differences between rank based and value based robust gene markers, we have different approaches in constructing committees.

It is relatively simpler to build a classification committee from the value based robust gene marker set by picking several top genes from the ranking list.

Building a classification committee from the rank based robust gene marker set is more complicated because the genes from this rank based ranking system are associated with a time value. Therefore, we have two options in choosing committee members: rank based or rank/diversity based.

- Rank based: A gene will not be chosen unless all genes with higher ranks were chosen;
- Rank/diversity based: in this option, we need to consider the diversity of committee members as well as their individual performances. That is, a gene will not be chosen solely because of its rank level. The advantages of having more diversified members include: more robust in implementation, more reliable, and smaller size of committee. Here, by diversity, we mean we want all available ranking systems have their own representatives in the committee. Increasing diversity inside a committee has been adopted extensively to get highly accurate committee classifiers from less accurate individual classifiers.

We designed a recursive method to select these committee members. In this method, we limit the number of committee members to $\tau + 1$ and each member is a representative of a time value, either one of the τ time points or the whole time series. The method consists of the following steps:

1. We will maintain a list of all gene candidates and a list of current committee members. To initialize, let $AllGene=\{g_1, g_2, ..., g_n\}$ denote the set of all genes, let *CommitteeSet={}* and *CandGeneSet=AllGene-CommitteeSet.*

2. We pick a time value *t* that does not have a representative in the committee. If there is more than one time value that does not have a representative in the committee, we will pick according to the time order (viewing the entire time series as the first in the ordering).

3. We rank the genes in *CandGeneSet* using the rank for time *t*. Let g^t be the gene having the highest rank for time *t*. We will call g^t the representative of time t in the committee. Add g^t to *CommitteeSet* and remove g^t from *CandGeneSet.*
4. Go back to step 2 until all the time values have their representatives in the committee.

6.2.3. Experiments and Discussions

To examine the effectiveness of robustness ranking algorithm, we designed an experiment. In this experiment, 27 microarray data are treated as test data. Both rank based and value based robustness ranking results are tested in the classifiers and figure 6.1 shows the results.

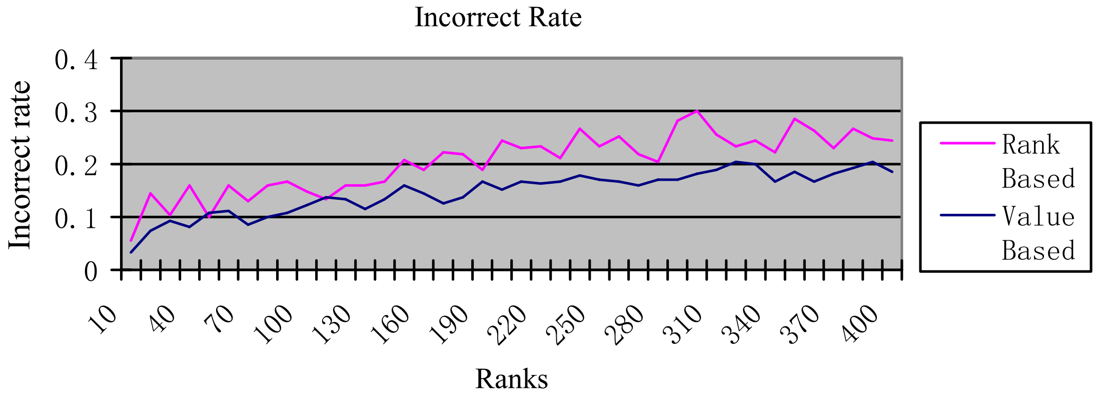

Figure 6.1: Ranking system comparison in microarray classification

There are only very limited number of genes that are ranked on the top of robustness ranking system and that can deliver a 100% accurate classification results. “Incorrect

Rate" in this figure stands for the ratio of misclassifications. The top 400 genes are divided into 40 consecutive groups with size of 10. For each group, the sum of wrongly classified cases is divided by all the available test cases (270). Clearly, an incorrect rate such as 0.5 is not acceptable.

From this experiment, we noticed three facts:

- As the robustness rank goes lower, the classification results get worse.
- Very few genes are perfect. We hope the committee approach can produce better performance.
- The average performance of value based robust gene markers is superior than rank based robust gene markers.

In order to improve the performance of fragment classification, we organize a committee for each robustness ranking system.

- For the value based robust gene marker set: genes 1511, 8600, 6717, and 1837 are chosen as the committee members simply because they are the top genes in this value based robustness ranking system. Then, we perform fragment classification by using this committee and the majority rule is applied. Experimentation shows that all 27 test microarray data are classified correctly.

- For the rank based robust gene marker set: genes 2324, 8600, 4453, and 1511 are chosen as the committee members by using the recursive method we introduced previously. Then, we do fragment classification by using this committee and the majority rule is applied. Experiment shows that all the 27 test microarray data are classified correctly.

The results show that the two committees all give quite satisfactory classification performance.

In our research, you may have noticed that we used only one column of the training data as the fragment. In practice, situations can be more complicated such as when more than one column is available. For example, two fragments, namely A and B, can be sampled at the same or different time from the same tissue. So, A and B may or may not have the same time value.

- If they have the same time values, we can classify them separately and try to combine the classification results in different ways, such as majority rule or averaging.
- If they have different time values, it is very likely that the gap between these two sampling times is known. We can classify them separately and try to justify these two results with the known time gap.

6.3. Wildness and Wild Discriminative Genes

6.3.1. Motivation

As we discussed in earlier sections, there are genes that are highly uniformly expressed in one class (normal class) and vary greatly/unpredictably in the other class (exposure class).

We call this phenomenon ***Wildness*** and these genes ***Wild*** genes. Although this phenomenon has been known in previous studies such as [32], not enough attention has been paid to it. The occurrence of such phenomenon can be attributed to various factors, such as:

- Many cancerous cells are associated with elevated rates of somatic mutation [38].
- Some genes which are tightly regulated by certain genes in normal tissues may lose this kind of the regulation after the treatment/exposure.
- Some genes show highly different levels of expression in cancer cells because of the existence of multiple hidden subtypes of the cancerous class. (In other words, one cancer type may actually be the union of two previously unknown subtypes.) Even though one gene is highly consistently expressed inside either subtype, it can still be possibly considered as a wild gene overall.
- There are cases where the cancer cells in one cancer tissue, are actually impure, or are composed of two or more different sub-types.

Finding wild genes is helpful in cancer researches by discovering unknown cancer types and identifying subtle gene regulations. Because it was not the focus of previous research, in this chapter, we will develop a new measurement to capture these wild genes, use these genes in classification, and examine the performance in experiments.

6.3.2. Approaches and Applications

We have mentioned and developed several ranking algorithms in previous sections. But, none of them is appropriate in spotting wild genes because they are designed to pick out the genes that are highly uniformly expressed in every class and are highly differentiated between different classes. Here, we develop another algorithm in order to find wild genes according to their unique characteristics.

In this part, we have three sections. We will first define a formula in order to measure the wildness level of each gene. Then we will try to incorporate the top genes from the wildness ranking list into applications such as fragment classification. Finally, a classification committee will be constructed where we consider the diversity factor.

6.3.2.1. Wildness Ranking

In order to single out the genes that cause the wildness phenomenon, we need to understand certain characteristics of wild genes:

- Wild genes should be expressed very consistently in normal tissues. Therefore, wild genes should have a very small deviation value in normal group.
- Wild genes should be expressed very inconsistently in exposure tissues. Therefore, wild genes should have a relatively large deviation value in exposure group.
- Because we will consider how wild genes can be utilized in classifiers, we want wild genes to be discriminative as well. Therefore, we need the average values of these two groups to vary from each other as much as possible.

Therefore, according to the characteristics we revealed above, we define the following formula to measure how wild and discriminative a gene is:

$$Wildness(g) = \left|Average_N(g) - Average_P(g)\right| \times \frac{Deviation_P(g)}{Deviation_N(g)}$$

From this formula, we can see that genes that have large wildness values normally have large deviation values in the exposure group (positive), small deviation values in the normal group (negative), and a big difference between the average values of these two groups.

Then, we rank all the genes by their *Wildness* value from high to low. The larger the wildness value a gene has, the wilder this gene is.

6.3.2.2. Classification Using Wild Genes

After the top genes are selected according to the wildness value, we will try to use them in fragment classification. Here, we will introduce a simple classification approach using wildness ranking information and deviation value. This method is an alternative approach to the one using robust genes.

The following figures show our ranking and classification strategy.

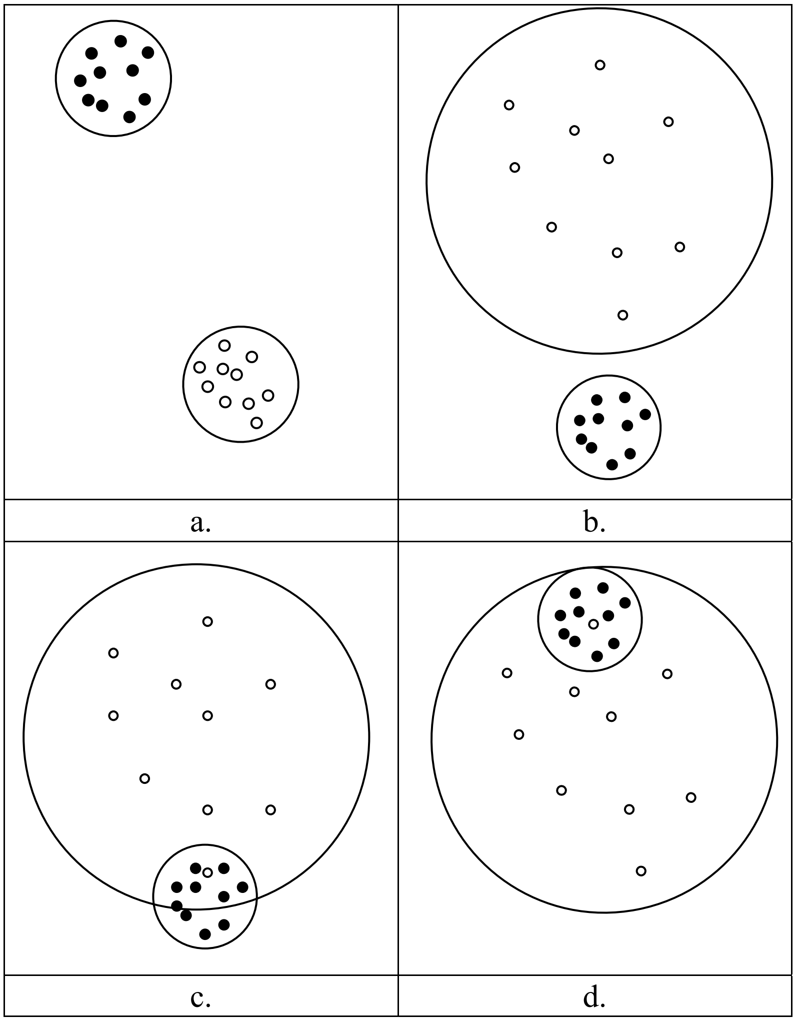

Figure 6.2: The idea of wild genes and classification
a: Gene selections preference in previous ranking methods
b, c, d: Gene selections preference in wildness ranking methods

In figure 6.2, the solid circles stand for samples from the negative class and the open circles stand for samples from the positive group. Figure 6.2.a shows what kind of genes we prefer in previous ranking methods: small variance for each group and far from each other. Figures 6.2.b, c, and d show our strategy in finding wild genes: small variances in the negative group and big variances in the positive group. The distance between the two groups is not as important as before. Apparently, Figure 6.2.d is the worst case in this example because of the overlap between the two groups.

The key idea of this approach is that when given a gene with great wildness value, it normally has a quite small standard deviation in the normal class. Our classification criteria is that when given a test microarray, if the expression value of the wild gene falls into certain narrow range around the average value of this gene in the normal group, e.g.

$NegRange = [\, Average_N(g) - K \times Deviation_N(g),\ Average_N(g) + K \times Deviation_N(g)\,]$,

it is quite likely that this test tissue is normal. Otherwise, it is quite likely that the test tissue is an exposed tissue. Here, the width of *NegRange* is $2 \times K \times Deviation_N(g)$. *K* is a parameter that determines the width of *NegRange* and its value has a great impact on the classification results.

The performance of the classification is very sensitive to the value of *K*. To find the most appropriate value, two factors have to be considered: accuracy and the possibilities of false positive/negative. Theoretically speaking, if the control group data set is normally distributed, according to the empirical rule, setting *K*=2 or 3 should deliver the most accurate results. However, *K* has to be adjusted for false negative and false positive cases.

- False positive: A sample that is negative and is classified falsely as a positive sample.
- False negative: A sample that is positive and is classified falsely as a negative sample.

Intuitively speaking, because the gene is wildly expressed in the positive group, the wider *NegRange* is, the more likely the value falls into the *NegRange*. When the width of *NegRange* is too big, some samples (including cancer samples) will be misclassified as negatives (false negative). If the width of *NegRange* is too small, some samples, including healthy samples, will be misclassified as positives (false positive). As a result, we cannot simply choose K=2 or 3. If we do, the possibilities of false negatives may be much higher than the possibilities of false positives. Normally speaking, this situation should be avoided because the price of false negatives and false positives can be different. For example, in cancer diagnoses, the price of false positives is much higher than that of false negatives. So, the best value of *K* is application dependant. Here, we only discuss how to experimentally determine the value of *K* if we want to find a *NegRange* such that misclassification possibilities are balanced for either group. Our experiments show that *K*=1.2 or 1.5 is better.

Also, we noticed that the number of wild genes in the ranking list that can be used in the classifiers is quite limited because if a gene's deviation is too large, the *NegRange* may become so wide that the classifiers won't make any sense at all.

6.3.2.3. Diversified Committee

It is quite obvious that the classification performance when using wild genes will not be as good as using robust genes. The reason is: since the expression values of tissues from exposure group are highly diversified, some may fall into the *NegRange*. However, it does not stop the wildness approach from being an alternative choice when robust genes are not available. The accuracy problem can be solved by constructing a committee. More importantly, the wildness concept can be of significant interest to the biological scientists.

It has been widely agreed (such as [6], [13], [45]) that to make the classification more accurate and powerful, committee members have to be organized in a diversified manner. Diversity measures have not been systematically studied until [35]. In [35], two methods are mentioned and ten measures are studied to measure how diversified the committee members are.

In our research, wildness is a new concept and wild genes are chosen according to the formula we put forward in earlier sections. But, wild genes are different from each other in that some wild genes can be randomly expressed or they may have a relationship such as regulation. As a result, having two genes that are closely related in the same committee should be avoided for two reasons: eliminate repeated votes and achieve better parsimony.

Because the number of committee members is limited, we will use the pair-wise approach mentioned in [35] to study the diversity of the committee. As to the measures, we will try to use two measures according to the following table:

	D_k correct (1)	D_k wrong (0)
D_i correct (1)	N^{11}	N^{10}
D_i wrong (0)	N^{01}	N^{00}

Table 6.4: Top genes from ranking results by using wildness value

Here, D_i and D_k are two different classifiers. N^{11} stands for the number of test cases that are correctly classified by both D_i and D_k; N^{01} stands for the number of test cases that are correctly classified by D_k but not D_i; N^{10} stands for the number of test cases that are correctly classified by D_i but not D_k.; N^{00} stands for the number of test cases that are incorrectly classified by both D_k and D_i.

- The Q statistics was introduced in [56] and implemented in [35], [46], and [25]. The Q value for these two classifiers is defined as

  $$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

  From the formula, we can easily see that the Q value should be between -1 and 1. If the two classifiers are totally independent, the expected value of Q should be 0. If the two classifiers tend to give the same classification results, the Q value tends to be positive. Otherwise, the Q value tends to be negative. We will use the Q value to determine how different the two classifiers are and how diversified the classification committee members are.

- However, our experiments show that the formula is not designed perfectly (we will explain this claim in the experiments). We alternatively used another measure called disagreement measure which was studied in [25] and [46]. The measure is given as:

  $$Dis_{i,k} = \frac{N^{01} + N^{10}}{N^{11} + N^{00} + N^{01} + N^{10}}$$

  From this formula, we see that the *Dis* value varies between 0 and 1. It tends to be closer to zero if the two classifiers are similar in classification results and tends to be closer to 1 if the two classifiers have less agreement in classification. After the comparison in our experiments, we will use this value to examine how similar two classifiers are in terms of classification results and try to diversify the committee members based on this measurement.

6.3.3. Experiments and Discussion

6.3.3.1. Wildness

The following two tables show us the top 10 and bottom 10 wild genes from the ranking results ordered by *Wildness* value. The standard deviation values for both control class and exposure class are shown as well.

Wildness Ranking	Gene name	$Deviation_N$	$Deviation_P$	$Average_N$	$Average_P$	*Wildness*
1	6491	0.0097	0.2866	0.0123	0.2969	8.401
2	1624	0.0200	0.3562	0.0278	0.3436	5.624
3	637	0.0141	0.2732	0.0212	0.2762	4.918
4	2803	0.0181	0.2833	0.0301	0.2701	3.753
5	7447	0.0313	0.4147	0.0525	0.3272	3.637
6	6315	0.0201	0.3074	0.0399	0.2752	3.590
7	8526	0.0303	0.3880	0.0405	0.2895	3.181
8	2022	0.0280	0.3731	0.0253	0.2511	3.006
9	7070	0.0469	0.4683	0.0407	0.3247	2.831
10	5080	0.0283	0.3627	0.0519	0.2694	2.784

Table 6.5: Top genes from ranking results by using wildness value

Wildness Ranking	Gene name	$Deviation_N$	$Deviation_P$	$Average_N$	$Average_P$	*Wildness*
8795	8794	0.2684	0.1797	0.3295	0.3294	0.0000
8796	6261	0.3084	0.2677	0.2834	0.2834	0.0000
8797	4078	0.3108	0.1946	0.2749	0.2750	0.0000
8798	1879	0.2842	0.2360	0.4339	0.4339	0.0000
8799	8539	0.2601	0.2661	0.3684	0.3684	0.0000

Table 6.6: Bottom 5 genes from ranking results by using wildness value

6.3.3.2. Fragment Classification Using Wild Genes

Then, another fragment classification experiment is conducted just like previous sections. We are using the top genes from the wildness ranking list as the features. According to the formula, four different parameters, *K*=1.0, 1.2, 1.5, and 2.0, are used and each of them produced a series in figure 6.3.

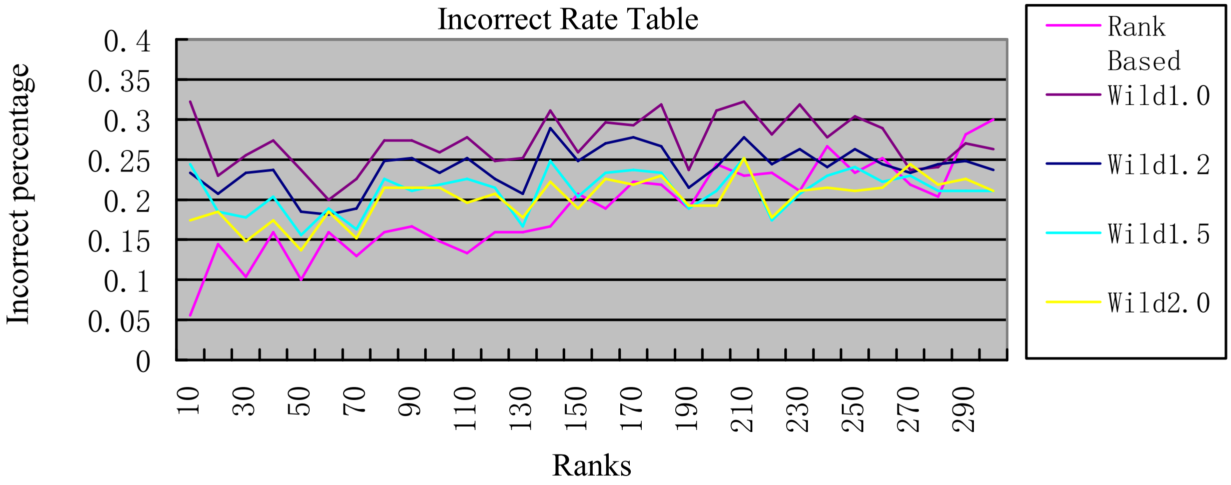

Figure 6.3: Ranking system comparison in microarray classification

After studying this figure, we can draw some conclusions:

- Different values of parameter *K* lead to different classification accuracy. Curves with *K* =1.5 and *K* =2.0 deliver similar performance in terms of accuracy. Both of them are significantly more accurate than the curve with *K* =1.2 and 1.0. The curve with *K* =1.0 is the worst among the 4 curves.
- Both rank based fragment classification and value based fragment classification outperformed wildness based fragment classification.
- These wild genes give very constant performance.

Next, we need to compare the ratio of false positive cases and false negative cases so that we can find the most appropriate value for parameter *K*.

Wildness Ranking	Gene Name	*Deviation$_N$*	K=1.0		K=1.2		K=1.5		K=2.0	
			FP	FN	FP	FN	FP	FN	FP	FN
1	6491	0.0097	5	0	4	1	1	2	0	2
2	1624	0.0200	4	3	2	3	1	5	1	5
3	637	0.0141	4	0	3	0	2	0	0	0
4	2803	0.0181	8	2	4	2	2	2	0	2
5	7447	0.0313	8	4	4	6	0	6	0	8
6	6315	0.0201	4	3	2	3	2	4	1	5
7	8526	0.0303	2	8	3	8	1	8	0	8
8	2022	0.0280	3	8	2	8	2	8	1	8
9	7070	0.0469	5	8	3	8	2	8	0	8
10	5080	0.0283	4	4	3	6	2	6	1	8

Table 6.7: False positives and false negatives in wildness classification

From table 6.7, we can easily see that with the same K value, the ratios between FN (false negative) over FP (false positive) are consistently increasing when the deviation value of normal group grows. For our dataset, setting K=1.2 or 1.5 may be the best value for the top genes in the wildness ranking.

6.3.3.3. Organizing Wild Genes Committee

Because the number of satisfactory wild genes in classifications is limited and the overall performance is dragged down by other genes, we manage a committee using the first several genes in the wildness ranking list to enhance the performance. In the next experiment, we used the top 5 genes from the wildness ranking list and the classification results show that for all the 27 test cases, 25 are correctly classified, 2 are mistakenly classified and none are unclassified.

We also considered the diversity of the committee in order to make the committee more accurate and simplified as well as avoid having too many similar classifiers in terms of accuracy. Q statistics and Disagreement statistics are evaluated and compared. Table 6.8 shows the N values which can be used to calculate the two statistics.

Wild gene ranking	Gene	1	2	3	4	5
1	6491	N/A	0, 3, 2, 22	0, 3, 6, 18	2, 1, 2, 22	3, 0, 3, 21
2	1624	N/A	N/A	0, 2, 6, 19	0, 2, 4, 21	0, 2, 6, 19
3	637	N/A	N/A	N/A	0, 6, 4, 17	0, 6, 6, 15
4	2803	N/A	N/A	N/A	N/A	2, 2, 4, 19
5	7447	N/A	N/A	N/A	N/A	N/A

Table 6.8: Ranking system comparison in microarray classification
(numbers are listed in the sequence of $N^{00}N^{01}N^{10}N^{11}$)

Wild gene ranking	Gene	1	2	3	4	5
1	6491	N/A	-1	-1	0.9130	1
2	1624	N/A	N/A	-1	-1	-1
3	637	N/A	N/A	N/A	-1	-1
4	2803	N/A	N/A	N/A	N/A	0.6521
5	7447	N/A	N/A	N/A	N/A	N/A

Table 6.9: Q statistics for pair-wised classifiers

Wild gene ranking	Gene	1	2	3	4	5
1	6491	N/A	0.1851	0.3333	0.1111	0.1111
2	1624	N/A	N/A	0.2962	0.2222	0.2962
3	637	N/A	N/A	N/A	0.37037	0.4444
4	2803	N/A	N/A	N/A	N/A	0.2222
5	7447	N/A	N/A	N/A	N/A	N/A

Table 6.10: Disagreement statistics for pair-wised classifiers

From the Q statistics table above, we can easily see that it has a major flaw. The goal of such a formula is to have the Q value close to 1 if the two classifiers tend to give the same classification and have the Q value be negative if they tend to disagree with each other. However, according to its formula, once N^{00} is equal to 0, the Q value will definitely be a non-positive number regardless how similar the two classification results are. And this is the reason why we want to try the Disagreement statistics formula.

From the Disagreement statistics table above, it is obvious that 6491 and 2803, 6491 and 7447 are fairly similar in classification results. So, we will remove gene 7447 from the committee. As a result, there are only four members in the committee and the classification is redone. Results show that 25 are correctly classified, none are mistakenly classified and 2 are unclassified. The results are better than before and the number of committee members has been decreased.

6.4. Time Recovery

6.4.1. Motivation

In previous sections, we discussed how to select the time-insensitive informative genes and wild genes and how to use them in fragment classifiers to determine the class label of the test samples without time value.

Time recovery is considered as the further development of the fragment classification. In this section, we introduce an algorithm to discover the time characteristic of a fragment *MTSO* by using interpolation after predicting its positive class label. To be more specific, when given a microarray, *X(t)*, if it is from the cancerous group, we use interpolation method to predict its time value, *t*. Here again, the time value, *t*, can be any value in a certain range. At the end, experiments are conducted to examine the performance of our algorithm.

6.4.2. Approach

Interpolation is a method that can be used to make predictions within the range of the dependant variable from the sample data so that a model can be generated. The formal definition of interpolation is:

Given a set of *n* pairs (X_k, Y_k) of numbers, we need to define a function, *f*, so that $Y_k = f(X_k), k = 1,2,...,n$. Such a function, *f*, is called the interpolation function. Interpolation functions have many different kinds, such as linear interpolation, polynomial interpolation, spline interpolation …

We choose linear regression to solve this time recovery problem. Linear regression is called "linear" because the relationship of the response to the explanatory variables is assumed to be a linear function. The reason we choose linear regression is that we want our approximation curve to be a monotonic function to ensure that each gene expression value gives exactly one time point in the estimation results. A monotonic function is a function that the dependent value (expression value) increases or decreases along with independent value (time) throughout the time series. Otherwise, when given one dependent value, there might be multiple time points that correspond to it, which leads to ambiguity.

We have to find the most appropriate genes for the interpolation. These genes need to have the following characteristics:

- Gradually: we want the expression values of such a gene to change smoothly throughout the time series (or differentiable, no sudden jumps).
- Constantly: we want the expression values of such a gene to change at some constant rate throughout the entire time series.
- Significantly: we want the expression values of such a gene to change at a high rate throughout the whole time series so that it can be more error resistant when used for interpolation.

In order to find these appropriate genes, several steps must be followed:

1. For a given set of microarray data from exposure group, we perform linear regression according to their time and expression values. The standard notation of linear regression for the *MTSO* data is: when given a time series for gene, g_k, the data are pairs of an independent variable (time *t*), and a dependent variable (expression value, *X*), $((t_i, X(g_k, t_i)) : i = 1,2,...n)$. To achieve the *LSE* (Least Square Error), which is defined as $\sum_{i=1}^{n}(\hat{X}(g_k, t_i) - X(g_k, t_i))^2$, the fitted line is written as:

$$\hat{X}(g_k, t_i) = b_1(g_k) + b_2(g_k)t_i$$

Here, $\hat{X}(g_k, t_i)$ is the predicted value obtained by using the equation. The regression equation is listed below:

$$b_2(g_k) = \frac{\sum_{i=1}^{\tau}((t_i - \bar{t})(X(g_k,t_i) - \overline{X}(g_k,t)))}{\sum_{i=1}^{\tau}(t_i - \bar{t})^2},$$

$$b_1(g_k) = \overline{X}(g_k,t) - b_2(g_k)\bar{t}$$

Here, $\bar{t}$ and $\overline{X}(g_k,t)$ stand for the mean values, $b_2(g_k)$ stands for the slope, and $b_1(g_k)$ stands for the intercept.

2. We use the following formula to calculate the interpolation score *(IPScore)* for each gene, g_k.

$$IPScore(g_k) = \frac{|b_2(g_k)|}{MSE(g_k)}$$

and we prefer genes with large *IPScore* values. In other words, we want our regression model for the selected genes to be steep (big $|b_2(g_k)|$) and approximated well (small Mean Squared Error, $MSE(g_k)$).

We rank all the genes according to their *IPScore* in the order from large to small.

From the *IPScore* ranking results, we choose the top genes so that we can evaluate the time value for the given test microarray data using the following formula:

$$\hat{t} = \frac{X(g_k,t) - b_1(g_k)}{b_2(g_k)}$$

We must emphasize that the interpolation results should have a limited effective range. In our experiments, all our original data is from 1 to 8 hours. The interpolation results at 10 hours should be voided.

In certain applications, especially when a single feature is not accurate enough, several genes can be selected according to *IPScore* ranking results. In such a committee, each gene is used to give an estimation and the average value can be calculated as the final prediction value.

6.4.3. Experiments and Discussion

The following table shows the *IPScore* ranking list for the exposure group.

IPScore Ranking	Gene name	b_2	b_1	*IPScore*
1	3686	0.1128	-0.0181	0.4973
2	5079	0.1119	-0.0226	0.4502
3	6669	-0.1050	0.9285	0.4433
4	7897	0.1142	-0.0582	0.4384
5	7171	0.1222	-0.0801	0.4378
6	7009	0.1237	-0.1102	0.4370
7	2028	0.1218	-0.1218	0.4357
8	1458	0.1242	-0.0447	0.4175
9	248	0.1156	-0.0254	0.4109
10	591	0.1218	-0.0965	0.4089

Table 6.11: Exposure group *IPScore* ranking list

Using the genes from table 6.11, we are able to interpolate the time value for a given microarray. The next table shows the interpolation results.

From table 6.12, we find out that the interpolation values are somehow close to the actual value, which means the interpolation algorithm works as we hoped even though improvements on accuracy are necessary.

To improve the performance, we choose first 3 genes from top of the ranking list and calculate the average prediction value of each interpolation results as the final estimation.

The results from the experiment are shown in following table. Improvements can be observed and further improvements can be done by removing outliers.

Expression value	Actual time value	Interpolation result
0.155	1	1.540
0.076	1	0.841
0.148	1	1.479
0.097	1	1.024
0.320	4	2.996
0.477	4	4.394
0.396	4	3.674
0.364	4	3.384
0.791	8	7.172
0.920	8	8.316
0.902	8	8.156
1.000	8	9.018

Table 6.12: Example interpolation when using gene 3686

Actual time value	Interpolation using 3686	Interpolation using 5079	Interpolation using 6669	Average
1	1.540	0.202	0.390	0.710
1	0.841	0.851	0.452	0.714
1	1.479	1.068	0.821	1.122
1	1.024	1.502	1.344	1.290
4	2.996	3.825	4.752	3.857
4	4.394	3.586	4.229	4.069
4	3.674	4.717	4.786	4.392
4	3.384	4.529	4.984	4.299
8	7.172	6.984	7.631	7.262
8	8.316	7.243	8.141	7.900
8	8.156	9.137	7.973	8.422
8	9.018	8.352	7.074	8.148

Table 6.13: Example interpolation when using gene 3686, 5079, and 6669.

7. CONTRIBUTIONS AND FURURE WORK

In this chapter, we will first conclude this dissertation by summarizing our contributions and then identify several possible future research directions for *MTSO* data analysis together with some preliminary work.

7.1. Contributions

MTSO data analysis is a relatively new research topic and the importance has not been fully recognized. Current practice shows the need of a systematic analyzing approach. In this dissertation, we developed several approaches to meet the classification needs. We used microarray data throughout our research.

The main opportunities and challenges associated with the *MTSO* data include: (i) each feature (gene) is associated with a time series, and (ii) there are many features to consider. How to identify and take advantages of these unique characteristics of time series is the focus of our research.

- *GST* Ranking and microarray classification: We started by studying a previous feature selection method called *GS* ranking algorithm for bi-class data without time. After pointing out that these features are time dependant, we used the *GST* method to rank genes within a single time point and we studied the performance of microarray classification using this method for feature selection. Our experiments show that the performance is quite satisfactory if the test data and sampling data are from the same time domain. However, the performance is not acceptable in the cross time domain scenario, which implies that the ranking results are time sensitive.

- *ET* Ranking and *MTSO* classification: We extended the *GST* method to utilize information extracted from entire time series and use top genes from *ET* ranking results in *MTSO* classification. Our experiments show that *ET* ranking results are quite accurate in *MTSO* classification. Experiments also show that *GST* ranking results are helpful in *MTSO* classification even they are not as superior as *ET* ranking results.
- Robustness and fragment classification: Fragment classification is a new classification challenge. In fragment classification, the test data is just a partial *MTSO* data instead of the whole. Also the time value of such a fragment data is missing. Our approach to solve such an issue is to find time insensitive features and use these robust genes in classification. We developed two different ranking systems either based on values or rankings. At the same time, in order to improve the classification accuracy, we organized decision committee in a diversified manner.
- Wildness and fragment classification: We noticed that some genes are very uniformly expressed in one class and are highly diversified in the other class. We call these features wild genes. In addition, metrics are designed to filter out wild

genes and used them as an alternative approach to robustness in fragment classification.

- Time recovery: As a natural extension of fragment classification, we use interpolation approach to estimate the time value of the partial data. Committees are organized to improve the prediction accuracy.

7.2. Discussions and Future Work

We will end this dissertation by pointing out several possible new research directions.

7.2.1. GST Ranking Changing Pattern

We know that gene expression values change from time to time. Biological scientists such as [24], [7], and [43] believe that the identification of genes with certain value change patterns is useful. [7] and [43] mainly focused on the identification of immediate-early responsive gene changes. But, authors in [24] pointed out the fact that late responsive genes, such as genes that are ultimately responsible for long-term changes, are just as important as others. They gave an example that induction of immediate-early genes in response to neuronal activity is responsible for setting the stage for long term changes in synaptic function. Their studies focused on the gene expression value changes in different stages. They used the *MTSO* data to recognize genes that are: early-response genes, late-response genes, constant-response genes, intermediate-response genes, and gradual-response genes, etc.

Value change patterns can be easily evaluated based on either z-ratio or fold-ratio [12].

Fold-ratio for the gene expression values is the ratio between controlled and exposed data. Here is an example using fold-ratio for late-response genes: during the testing period, if the fold-ratio stays around 1 at the early stage and change drastically at the end, we will call such genes late-response genes.

But, the fact most researchers have ignored is, according to the *GST* ranking results in our experiments, not just the expression values, the ranks of a gene change from time to time as well. For a given gene, at some time points, the ranks can be very high which means it could be an informative gene that is highly correlated with its class; but, at some time points, the ranks could be really low which means it is not closely correlated with the class label. So, we naturally come up with a new idea: by using the gene ranking information, associate certain genes with time values:

- Immediate-early-stage discriminate genes. These genes are only suitable for classifying microarray data that are collected immediately or shortly after the exposure. In our example, gene 5 is a good candidate. At time point 1, it is ranked as number 1. After a while, it is ranked as number 7. In our experiments, gene 7070 is one of such genes.
- Intermediate-stage discriminate genes. These genes are suitable for classifying microarray data that are collected in the middle of the experiments after the exposure. In our example, gene 6 is a good candidate. At time point 2, it is ranked as number 1. It is ranked as number 7 at other time points. In our experiments, gene 2510 is one of such genes.
- Late-stage discriminate genes. These genes are only suitable for classifying

microarray data that is collected long after the exposure. In our example, gene 7 is a good candidate. At time point 1 and 2, it is ranked quite low. But, at the end of the experiment, it is ranked as top 1. In our experiments, gene 8468 is one of such genes.

- All-stage discriminate genes. These genes are suitable for classifying microarray data that is collected any time after the exposure. In our example, gene 2 and gene 4 belong to this group. Even though neither of them is ranked at the top at any time points, they are ranked pretty high at every time points. In our experiments, gene 5519 is one of such genes.

For some genes, the value changes and the rank changes may have the same pattern. But, for most genes, it may not to be true. Recognizing these genes could be helpful in further revealing the genes' working mechanism after exposure.

7.2.2. Multi-Class Classification

After the algorithms for bi-class classification are developed, the natural idea for the next step is: is it possible to adapt it for multi-class classification? The research topic is like: training data is obtained by treating each tissue with only one toxin, and testing data is obtained similarly. However, there is more than one toxin possibility. We need to determine whether a tissue is contaminated and by which toxin if the classification result is positive.

Problem definition: There are *n* types of cancers that are to be examined, which means this is a *n+1* classes classification problem: *n* cancer groups and 1 normal group. When given a sample, we need to identify its class label.

We designed the following approach. Because we do not have any appropriate dataset, it only represents some naïve thoughts that need to be further developed and examined.

1. For each cancer class, *k*, we pair it with the normal class and form a bi-class classification problem.
2. We extract the robust gene marker candidate sets from the bi-class classification. As a result, we have *n* robust gene marker candidate sets that each one is associated with a cancer group. The difference between robust gene marker candidate sets and robust gene marker sets we mentioned in the previous section is the size. The robust gene marker candidate sets have more genes at every time point than before.
3. Diversify the included genes from each robust gene marker candidate sets. Try to eliminate the multiple occurrences for the same gene in different sets. We call the results set for class *k* as Character Gene Sets (CGS_k), and associate the average values of these genes in class *k* as Character Value Sets (CVS_k). The number of genes in class *k*'s Character Gene Sets can be written as $SS(k)$. For a gene in the *k*'s Character Gene Sets $CGS_k(i)$, the average value is expressed as $CVS_k(i)$.
4. For a given sample, we determine its class label according to its gene expression values.
    a. We match the new sample to the existing *CGS*es. Because we have *n* *CGS*es, we need to match *n* times. The matching procedure uses Euclid distance, and gives confidence value as the result. $GV(CGS_k(i))$ is the

expression value of gene *CGS_k(i)* in the test sample.

$$Confidence(k) = \frac{\sum_{i=0}^{SS(k)} | CVS_k(i) - GV(CGS_k(i)) |}{SS(k)}$$

b. We compare the confidence levels among the *n* matching results. If the comparison with Character Gene Sets *k* gives the highest confidence value and it is above a certain threshold, we label the given sample as class *k*. Otherwise, we assign it as a normal sample.

7.2.3. Other Possible Topics

There are several other research topics that remain open:

- Training data is obtained by treating each tissue with one toxin or several toxins, and testing data is obtained similarly. The problem is to determine whether a test tissue is contaminated and by which toxins if the classification result is positive.
- Training data obtained by treating each tissue with one toxin, and testing data obtained by treating each tissue with one toxin or several toxins. The problem is to determine whether a test tissue is contaminated and by which toxins if the classification result is positive.

REFERENCES

[1] H. Andre-Jonsson and D. Badal. Using Signature Files for Querying Time-Series Data. *In Principles of Data Mining and Knowledge Discovery*, *Trondheim, Norway,* 211-220, June 1997.

[2] Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D. & Levine, A. J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Nat. Acad. Sci. USA 96,* 6745–6750.

[3] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. *In Proc. of the Fourth Int'l Conference on Foundations of Data Organization and Algorithms, Chicago*, October 1993.

[4] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, Z. Yakhini, Tissue classification with gene expression profiles. *In Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, Tokyo, Japan,* 2000.

[5] Beissbarth,T., Fellenberg,K., Brors,B., Arribas-Prat,R., Boer,J., Hauser,N.C., Scheideler,M., Hoheisel,J.D., Schutz,G., Poustka,A. and Vingron,M. Processing and quality control of DNA array hybridization data. *Bioinformatics*, 11, 1014–1022.

[6] Eric Bauer, Ron Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning,* vol. 36, 105-139, 1999.

[7] Brakeman, P. R., Lanahan, A. A., O'Brien, R., Roche, K., Barnes, C. A., Huganir, R. L. & Worley, P. F. Homer: a protein that selectively binds metabotropic glutamate receptors. *Nature,* 386, 284–288, 1997.

[8] Breiman, L. Bagging Predictors, *Machine Learning*, Vol. 24, No. 2, 123-140.

[9] Raymond J. Cho, Michael J. Campbell, Elizabeth A. Winzeler, Lars Steinmetz, Andrew Conway, Lisa Wodicka, Tyra G. Wolfsberg, Andrei E. Gabrielian, David Landsman, David J. Lockhart, and Ronald W. Davis. Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle**.** *Cell*, Vol 2, 65-73, July 1998.

[10] K.-P. Chan and A.-C. Fu. Efficient time series matching by wavelets. *In ICDE,* 1999.

[11] Ting Chen, Vladimir Filkov, Steven S. Skiena. Identifying gene regulatory networks from experimental data. *Parallel Computing,* Volume 27, Issue 1-2 January 2001.

[12] Cheadle, C., Vawter, M. P., Freed, W. J., and Becker, K. G. Analysis of microarray data using z score transformation. *Journal Mol Diagn,* 5, 73-81, 2003.

[13] Drucker, H., Cortes, C., Jackel, L., LeCun, Y., & Vapnik, V. Boosting and other ensemble methods. *Neural Computation,* 6, 1289–1301, 1994.

[14] M Dash, H Liu. *Intelligent Data Analysis,* 1, 131–156, 1997.

[15] Dudoit,Y., Yang,Y.H., Callow,M.J. and Speed,T.P. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Technical Report 578, Department of Statistics, UC Berkeley, CA, USA.*

[16] Everitt, B. *Cluster Analysis, third edition, Edward Arnold, London,* 1993.

[17] Eickhoff,B., Korn,B., Schick,M., Poustka,A., and Van Der Bosch,J. Normalization of array hybridization experiments in differential gene expression analysis. *Nucleic Acids Res*., 27, e33, 1999.

[18] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *In Machine Learning: Proceedings of the Fifteenth International Conference,* 1998.

[19] Friedman, J. Another approach to polychotomous classification. *Technical report, Stanford Univ,* 1996.

[20] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Peer. Using Bayesian networks to analyze expression data. *Proc. 4th Annu. Int. Conf. Computational Molecular Biology (RECOMB'00),* 127-135.

[21] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *In Proc. of the ACM SIGMOD Conference on Management of Data,* May 1994.

[22] Filkov V, Skiena S, Zhi J. Analysis techniques for microarray time-series data. *Journal of Computational Biology.* 9(2):317-30, 2002.

[23] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, E. S. Lander. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, (286), 531-537, 1999.

[24] Hong SJ, Li H, Becker KG, Dawson VL, Dawson TM. Identification and analysis of plasticity-induced late-response genes. *Proc Natl Acad Sci US A.* Feb 17;101(7):2145-50, 2004.

[25] Ho, T. The random space method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:8, 832–844, 1998.

[26] Trevor Hastie and Robert Tibshirani, Classification by pairwise coupling. *The Annals of Statistics*, Vol. 26, No. 2, 451–471, 1998.

[27] E. Keogh, K. Chakrabarti, M. Pazzani, and Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems,* 2000.

[28] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *In Proc. of ACMSIGMOD*, 151-162, 2001.

[29] F. Korn, H.V. Jagadish and C. Faloutsos. Efficiently Supporting ad Hoc Queries

in Large Datasets of Time Sequences. In *Proc. ACM-SIGMOD International Conference on Management of Data*, 289-300.

[30] Kroll,T., Odyvanova,L., Clement,J.H., Platzer,C. , Naumann,A., Marr,N., Höffken,K. and Wölfl,S. Molecular characterization of four breast cancer cell lines by expression profiling. *J. Cancer Res. Clin. Oncol.,* 128, 125–134, 2002.

[31] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. *in Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97), D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, eds*., pp. 24--30, AAAI Press, 1997.

[32] A. Keller, M. Schummer, L. Hood, W. Ruzzo. Bayesian Classification of DNA Array Expression Data. *Technical Report,* August, 2000.

[33] J. Khan, et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural network*s. Nat. Med.* 7, 673-679, 2001.

[34] T. C. Kroll and S. Wölfl. Ranking: a closer look on globalisation methods for normalisation of gene expression. *Nucleic Acids Res.* Jun 1;30(11):e50, 2002.

[35] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning.* 51-2, 181 – 207, 2003

[36] Y. Lu, J. Han. Cancer classification using gene expression data. *Information Systems.* 28, 243-268, 2003.

[37] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. *Data Mining and Knowledge Discovery, Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery 2003*.

[38] Loeb, L. Cancer cells exhibit a mutator pheontype. *Advanced Cancer Research,* 72, 25-56, 1998.

[39] AV Oppenheim, RW Schafer, JR Buck. Digital Signal Processing. *Englewood Cliffs, New Jersey, Prentice-Hall*, 1975

[40] CH Ooi, P Tan. Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. *Bioinformatics,* 19(1), 37-44, 2003.

[41] S Peng, Q Xu, XB Ling, X Peng, W Du, L Chen. Molecular classification of cancer types from microarray data using the combination of genetic, *FEBS Lett,* 2003.

[42] John Quackenbush. Microarray data normalization and transformation, *Nature Genetics,* 32, 496 – 501, 2002.

[43] Qian, Z., Gilbert, M. E., Colicos, M. A., Kandel, E. R.&Kuhl, D. *Nature,* 361, 453–457, 1993.

[44] D. Rafiei, and A. Mendelzon. Efficient retrieval of similar time sequences using DFT. In *Proc. of the 5th Intl. Conf. on Found. of Data Org. and Alg. (FODO '98), Kobe, Japan,* November 1998.

[45] Schapire, R. Theoretical views of boosting. *In Proc. 4th European Conference on Computational Learning Theory.* 1–10, 1999.

[46] Skalak, D. The sources of increased accuracy for two proposed boosting algorithms. *In Proc. American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop.*

[47] R. Schapire. A brief introduction to boosting. *In Proceedings of International Joint Conference on Artificial Intelligence,* 1401--1405, 1999.

[48] Schuchhardt,J., Beule,D., Malik,A., Wolski,E., Eickhoff,H., Lehrach,H. and Herzel,H. Normalization strategies for cDNA microarrays. *Nucleic Acids Res*., 28, e47, 2000.

[49] Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, Eisen MB, Brown PO, Botstein D, Futcher B. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*

9(12):3273-97, 1998.

[50] D. Slonim, P. Tamayo, J. Mesirov, T. Golub, & E. Lander. Class prediction and discovery using gene expression data. *Proceedings of the Fourth International Conference on Computational Molecular Biology (RECOMB), Tokyo, Japan,* 263-272.

[51] David. M. J., Tax, and Robert. P. W., Duin. Using two-class classifiers for multiclass classification. *International Conference on Pattern Recognition.*

[52] Tseng,G.C., Oh,M.K., Rohlin,L., Liao,J.C. and Wong,W.H. Issues in cDNA microarray analysis: quality filtering, channel normalization, models of variations and assessment of gene effects. *Nucleic Acids Res.,* 29, 2549–2557, 2001.

[53] Tsymbal, A. Decision committee learning with dynamic integration of classifiers. *In ADBISDASFAA,* 265–278, 2000.

[54] Y. Wu, D. Agrawal, and A. Abbadi. A Comparison of DFT and DWT based Similarity Search in Time-Series Databases. *Proceedings of the 9th International Conference on Information and Knowledge*, 2000.

[55] Wang,X., Ghosh,S. and Guo,S.W. Quantitative quality control in microarray image processing and data acquisition. *Nucleic Acids Res.,* 29, e75, 2001.

[56] Yule, G. On the association of attributes in statistics. *Phil. Trans*., A, 194, 257–319, 1900.

[57] L.K. Yeung, H. Yan, A.W.C. Liew, L.K. Szeto, M. Yang, and R. Kong. Measuring Correlation between Microarray Time-series Data using Dominant Spectral Component, *Proceedings of the 2nd Asia-Pacific Bioinformatics Conference, Dunedin, New Zealand,* pp.309-314, APBC2004.

[58] Zien,A., Aigner,T., Zimmer,R. and Lengauer,T. Centralization: a new method for the normalization of gene expression data. *Bioinformatics,* 17(Supp. 1), S323–S331, 2001.

[59] H. Zhang, C. Yu, B. Singer, M. Xiong. Recursive partitioning for tumor classification with gene expression microarray data. *Proc. Natl. Acad. Sci.* USA 98(12), 6730–6735, 2001.

[60] James N. McDougal, Carol M. Garrett, Carol M. Amato, and Steven J. Berberich. Effects of Brief Cutaneous JP-8 Jet Fuel Exposures on Time Course of Gene Expression in the Epidermis. *Toxicological Sciences*, 95(2):495-510, 2007.