University of Alberta

FEATURE SELECTION IN MICROARRAY GENE EXPRESSION DATA ANALYSIS AND CONTAGIOUS VIRAL STRAIN COMPUTATIONAL GENOTYPING

by

©

Zhipeng Cai

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2008

Your file Votre référence
ISBN: 978-0-494-46289-8
Our file Notre référence
ISBN: 978-0-494-46289-8

Abstract

As a fast developing research field, bioinformatics and computational biology have brought about numerous amazing biological discoveries through computational approaches. The available large volume of genomic data provides a large amount of information, but also acquires computing techniques to extract the useful information hidden inside. This thesis focuses on data dimensionality reduction and information extraction via selecting the most informative features. The research projects concerned include gene expression microarray data analysis and computational genotyping of several contagious viral strains. In gene expression microarray data analysis, we regard genes as features and focus on identifying the most discriminative genes. We improve the existing classifiers and propose more efficient classification algorithms based on the biomarkers we selected. Our method has shown significant classification improvements on cancer microarray data. On viral strain genotyping, we adopt the complete composition vector representation for whole genomes and regard each nucleotide strings (and peptides) as genomic features. Afterwards, we apply the gene selection algorithms from the gene expression microarray data analysis to locate the most genotype specific strings and subsequently construct the genotypers. We also demonstrated success on various samples of virus data (HIV-1, FMDV, HCV and AIV). Our feature selection based algorithms have shown highly efficient performance on information extraction from large volume data, which were never achieved before either computationally or via wet-lab approaches. These methods may contribute to more biological discoveries and assist with clinical diagnosis if used properly.

Acknowledgements

First and the foremost, I would like to thank my thesis supervisors, Dr. Guohui Lin and Dr. Mohammad R. Salavatipour, for their patient instruction and generous support. They open one black box after another for me and give me theoretical and practical ability in problem formation and solving. They are also the ones who kept my work on the right track. I have learned from them not only the ability of doing research, but an attitude for doing research.

I would like to thank Dr. Randy Goebel, who has also provided a great deal of insights into this research. He contributed many good suggestions on the experiments.

Next, I would like to thank my family for their support in all phases of my life.

Also I would like to thank Dr. Lizhe Xu and all my colleagues, Xiang Wan, Gang Wu, Xiaomeng Wu, Jianjun Zhou, Yi Shi and Meng Song, for creative discussion. I really enjoyed the many chats with them that have enriched my view of algorithms.

Finally, I felicitate the cherish the opportunity to study in the Department of Computing Science. Here, I find myself studying in a distinguished environment encompassed by an academic atmosphere where originality is promoted and individual potential is tapped.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Biological and medical research has been one of the fastest developing research fields in the last two decades. With more and more biological data being used to provide molecular and genetic level understanding, almost all the research problems in biological and medical sciences are no longer easily solvable by phenotypical observations. Bioinformatics and computational biology is an emerging inter-discipline in which mathematical and computational techniques are applied to solve biological problems. It has since brought out numerous wonderful biological discoveries, which might not have been unraveled through traditional wet-lab based biological research methodologies. The revolution of biological technology has generated large volumes of genomic data, which acquire advanced computing techniques to extract the useful information hidden inside. Such knowledge discovery processes pose huge computational challenges in both computer memory consumption and CPU usage.

The shift of research focus from the protein level to the genetic level has made significant contributions to the acceleration of biological studies. Protein is an essential component of cells in a living organism. Not only in the structural elements of cells, but proteins also participate in metabolism, cell cycle, signaling, and other important cellular functions. Protein is the product of another cell component, named nucleic acid. Nucleic acid is the genetic material in the cell which has two common forms: deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). The functional unit of nucleic acid is called the gene. Initiated by gene expression, genes encode proteins and the control of protein synthesis. In the eukaryote, messenger RNA (mRNA) is synthesized in the nucleus according to the DNA sequence, a process called the *RNA transcription*. After that, protein is synthesized according to the mRNA in the cytoplasm, a process called *protein translation*. If the protein is the main executor of cellular functions, then the gene is definitely the commander and controller. As the determinant of protein, it is more sensible to study the cellular functions and the specific protein activity at a genetic level. The composition unit of the gene is the nucleotide. Under normal configuration, the DNA has a double helical structure, with two completely complementary strands winding together. In DNA, each nucleotide is associated with one of the four bases A, T, C and G. A DNA strand can be expressed as a strand with the sequential codes of A, T, C and G. Among the four bases, A is complementary to T, and C is complementary to G, which means base A will combine closely with base T, and the same for bases C and G.

Bioinformatic methods serve the purpose of solving biological and medical problems. One of the most challenging problems we face today is disease. From cancer and HIV, which has existed for a long time and are currently incurable, to the chicken flu and SARS, which appeared quickly but is extremely devastating, tremendous efforts have been spent on the genetic studies of their mechanism. The high-throughput genomic data not only provides us a good opportunity to detect and diagnose many diseases or aid in optimizing therapeutical options, but also brings many computational difficulties for the most avid explorer. Among the large number of potential features discriminating between disease and control samples compared, only a small number of them carry a biologically significant signal. Our research motivation is to identify informative features that are likely to provide useful piece of information related to disease as well as the relations among these features. Our research includes gene expression microarray data analysis and computational genotyping on several contagious viral strains, both of which have direct impacts on human and animal health care.

Although there are many machine learning methods proposed for feature selection, most of them are not suitable for the specific biological datasets. In this dissertation, we have proposed several feature selection methods for both cancer microarray data and virus strain data. These feature selection methods have been shown successful through extensive testing on real datasets.

The first type of data we target on is cancer gene expression microarray data. Unlike non-malignant benign tumors that grow in a self-limited manner, and do not invade or metastasize, malignant tumor cells have three major characteristics, aggressive (growing and dividing continually), invasive (invading and destroying neighboring tissues), and sometimes metastatic (spread to other parts of the body). Cancer affects people at all ages and there is evidence showing that the risk for the more common varieties tends to increase with age. As a major disease threatening human lives, cancer causes about 13% of all deaths. In the year of 2007, 7.6 million people died from cancer in the world [2].

Unlike virus-induced disease, most of cancer occurs on patients who have specific genetic characteristics. Recently, a tremendous amount of cancer research has been focusing on understanding the genetic mechanism and process in turning a normal cell cancerous. A number of genes have been identified as cancer-related and play a role in the development of certain types of cancers. Cancer genes can also be inherited [3]. Some people carry genes which make them more likely to develop cancer. Because the existence of "cancer gene", it is very obvious that studying cancer from patients' genetic profiles is very important. Due to the large size of the human genome, studying whole genome is technically impossible. Gene expression, as an alternative, is widely adopted. Another important reason to study the gene expression is because it carries information from the DNA to direct the protein synthesis. Messenger RNA represents the activity or functions of genes, rather than static genes. Genes vary from species to species, and from tissue to tissue in the same organism. Even in the same cell, the gene expression can be largely different due to different conditions, for example, different proteins will be synthesized under different conditions. The different gene expression thus provides precursory markers for cellular functions and activities.

In the past, collecting large numbers of gene expression values at the same time was technically impossible. The advent of the microarray technology changed this situation. Gene expression microarray is a technology to monitor the levels of thousands of genes simultaneously. It is also called DNA microarray, gene chip, or DNA chip. Compared with the traditional biological experiments, which can only examine one or a few genes at a time, such as northern blotting, the microarray technology provides us with an opportunity to understand the living cells better at a genetic level. Using microarray technology, the efficiency is increased thousands of times, especially when the sample volume is limited. The microarray technology is also superior in terms of its broad and systematic view at cellular activity compared with narrowly observing single genetic elements through traditional methods. By taking advantage of this tool, biological problems can be understood faster and more comprehensively.

DNA microarray is usually on a piece of glass slide, a silicon chip, or a nylon membrane. Different DNA segments are immobilized on the slide with specific positions. A few methods are being used to fabricate the microarray chip, depending on the type of the gene expression microarrays and the length of the probes (Figure 1.1). To name a few, fine-pointed pins onto glass slides, photolithography using pre-made masks, photolithography using dynamic micromirror devices, ink-jet printing, etc [4].

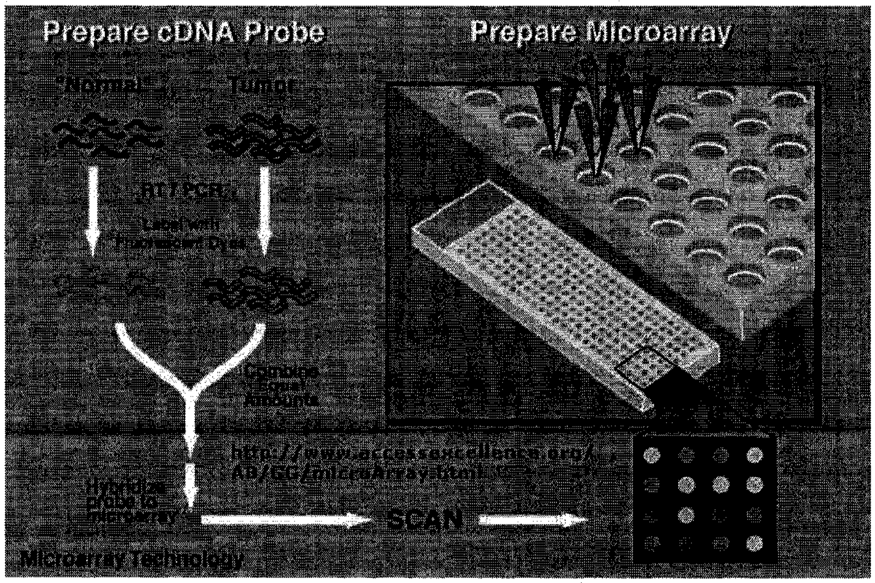

Prepare cDNA Probe
Prepare Microarray
"Normal"
Tumor
RT / PCR
Label with Fluorescent Dyes
Combine Equal Amounts
http://www.accessexcellence.org/AB/GG/microArray.html
Hybridize probe to microarray
SCAN
Microarray Technology

Figure 1.1: Microarray experiment procedure [1].

The DNA segments fixed on the slides are usually called probes. Different types of microarray are built with different types of probes. A cDNA microarray uses the cDNA fragments as probes, which are usually the complementary sequences of mRNA, made by polymerase chain reactions (PCR). The basic theory underlying the gene expression microarray experiment design is the DNA's complementary base pairing. On the microarray chip, only single DNA strands are immobilized. When the DNA chip is exposed to DNA samples (single strands) for hybridization, the complementary sample strand with a fluorescent label will hybridize to the probe and fix on that location.

Oligonucleotide microarray immobilizes part of the DNA sequence on the chip, usually with $20 - 80$ mers. Shorter oligonucleotide microarray is more possible for unspecific hybridization. One microarray experiment can test on one sample, or two mixed samples, which are known as one-channel experiment and two-channel experiment, respectively. Gene expression levels can be obtained from one-channel microarray experiments. Two-channel experiments are more often used to compare the gene expression levels between two samples [5]. As many as $30,000$ cDNAs can be included on the surface of a microscope slide. The microarray used in this study is the cDNA oligonucleotide microarray produced by the Affymetrix company. To eliminate the effect of unspecific hybridization for short oligonucletotide probes, Affymetrix adopts a specific probe design, called the Perfect Match/Mismatch (PM/MM) probe strategy. Their microarray chip has a second probe (MM) that is identical to the perfect match (PM) one except for a mismatched base placed close to the PM probe [6]. The MM probe serves as the background hybridization noise that can be subtracted from the PM probe signal to get the perfect hybridization [6].

The sample DNA strand is also called the target. The samples are all labeled with fluorescent dyes with a certain wavelength. Some commonly used dyes include rhodamine and fluorescein or Cy3 and Cy5 [5]. The more targets there are in the samples, the more intense the fluorescence is. Thus the intensity of the fluorescence represents the level of gene expression in the target samples. To display the intensity of the fluorescence, usually a laser will emit certain length of light to excite the fluorescence, and the fluorescence will display a visible light, for example, red or green. A camera will capture the light and send the image for processing. The probes are usually positioned orderly in the array. After the fluorescence screening, the microarray results are present in a matrix with certain rows and columns. Each cell in the matrix represents the expression level of a specific gene in a specific sample. In other words, let g_{ij} record the expression level of the i-th gene in the j-th experiment. Then $G_{m \times n} = (g_{ij})_{m \times n}$ is the expression matrix of m genes in n experiments (Table 1.1).

Normally a microarray chip contains thousands of genes, making the microarray dataset large. Although microarray is expected to be of high information content, some technical pitfalls still exist as bottlenecks to obstruct more insights to be discovered. How to overcome these difficulties and extract useful information from the large amount of data remains the central focus for the bioinformatics researchers. One frequent problem with microarray data analysis is the existence of missing

	$Sample_1$	$Sample_2$	$Sample_3$	$\cdots$	$Sample_n$
$Gene_1$	g_{11}	g_{12}	g_{13}	$\cdots$	g_{1n}
$Gene_2$	g_{21}	g_{22}	g_{23}	$\cdots$	g_{2n}
$Gene_3$	g_{31}	g_{32}	g_{33}	$\cdots$	g_{3n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$Gene_m$	g_{m1}	g_{m2}	g_{m3}	$\cdots$	g_{mn}

Table 1.1: Gene expression microarray data as a matrix, where g_{ij} is the expression value of the i-th gene in the j-th experiment.

values in a microarray dataset. A number of reasons could cause missing data, to name a few, insufficient resolution, image corruption, or even dust and scratches on the slide [96]. However, most gene expression data analysis algorithms, such as gene selection, classification, and network design, require the expression matrix to be complete, i.e. without any missing values. It is absolutely necessary to design efficient missing value estimation methods in order to ensure further dataset analysis. Many computational methods have been proposed to target this problem. In this thesis, we designed an efficient missing value estimation method, the *Iterated Local Least Square* (ILLSimpute) method, and demonstrate that ILLSimpute outperforms other existing missing value imputation methods.

Another common problem in microarray data is the unbalance between the sizes of gene sets and the sample pool. The high cost of microarray experiments makes it impossible to run experiments on a large number of samples. The huge number of genes versus a tiny number of samples wakes up the curse of dimensionality. The unbalance of dataset makes the applications in microarray more difficult. Identification of discriminatory genes is one such application. In general, among the thousands to tens of thousands of genes which are monitored simultaneously in multiple experiments, only a fraction of them are biologically relevant and can be identified as contributors to tissue samples' properties. These interesting genes are usually differentially regulated under experimental conditions, i.e, their expression levels are increased or attenuated, compared to the normal levels, and therefore these genes can be regarded as biomarkers. Identifying these biomarker genes, a process referred to as gene selection, is very important in many applications such as disease subtype discovery and genetic profiling [102, 16]. Other genes, such as house-keeping genes whose expression levels are largely unchanged under different conditions, are less important in providing information to downstream data analysis. Therefore, one of the major tasks for computational biologists is to find these feature discriminatory genes. On the other hand, one of the main purposes of gene selection is to build a good classifier to recognize the different sample classes. Clearly, high classification accuracy hints at the good quality of these discriminatory genes. Most current gene selection methods are suitable for the binary classification, while not applicable to the more complicated condition, multi-class classification. With the methods to distinguish normal and diseased types getting more and more mature, the focus is switched to the identification of cancer subtypes. Accordingly, the multi-classification methods are more desirable. Therefore, in this dissertation, we will address the more complicated multi-class problem. Several gene selection methods have been proposed for the purpose of multi-class classification. Note that, classification can not only be used for disease subtype prediction, but also for patients survival time prediction and patients drug response prediction, that is, it can be used on both diagnosis and prognosis. Microarray data can also provide information more closely related to clinical applications. For example, using genetic profiles, some models or classifiers can be devised to predict the cancer patients' survival time, the possibility of resistance to certain drugs, cancer subtype, etc. We can show that our gene selection methods work quite well on cancer disease subtypes prediction. We will expand the method to other subtypes prediction in the future work.

Another type of data we are interested in is virus whole genome dataset. Genomics is the study of an organism's entire genome. Genomic research focuses on identifying the entire DNA sequence of organisms and genetic mapping. Before the end of the last century, identifying genomic sequences was almost impossible economically. However, with the emerging new molecular technologies, such as PCR, the whole genomic study has made tremendous progresses. A good example will be the famous human genome projects. Within 13 years, the genomic profile of human beings were identified successfully.

The large genomic data provides us with an opportunity to study the viruses at the genetic level. Many human diseases are induced by viral infection. Some are contagious and even life-threatening. Viruses are notorious for their rapidly changing and unpredictable variations. The genetic diversity is challenging medical treatments both technically and time wise. Some viruses have many types and subtypes. Each type has its genetic and geographic characteristics, calling for different strategies against it. Meanwhile, new types keep emerging through mutation and recombination. Hence it is critical to subtype them accurately and identify their evolutionary relationships. It is worth knowing that some of the recombinations are so complicated, that the recombinants can not be simply thrown to traditional subtypes. This research subject is called phylogenesis, the study of evolutionary relatedness among various groups of organisms. Phylogenesis study has been applied to many viruses, such us AIV, HIV-1, HCV, and FMDV, etc.

There exist several methods attempting to address phylogenetic questions from a whole genome perspective, based on efficient information representation of the whole genomes while bypassing the high computational complexity stage of multiple sequence alignments [64, 38, 45, 71, 83, 23, 41, 89, 87, 40, 86, 88]. All of these approaches are intended to extract the hidden evolutionary information from the whole genomes, but from different angles. For example, gene content based methods [83, 82, 41, 42] mainly concentrate on a portion of homologous genes shared by multiple genomes, and then define an evolutionary distance between two genomes based on their gene sharing percentage. Alternatively, the compression based methods [64, 38, 23] generally regard the whole genomes as plain text, and define the similarity between two genomes as the relative compression ratio. The disadvantages of the above two approaches are that, first, the former requires prior knowledge on homologous genes and, second, the latter suffers from aggregate errors arising from compression.

In this thesis, we extend single nucleotide or single amino acid composition to study string composition for whole genomes where a string is a consecutive segment of nucleotides or amino acids. There are plenty of composition strings and not every composition string contributes equally to the evolutionary distance calculation. Finding those feature strings is important for virus subtyping and virus recombination prediction. These feature composition strings can be regarded as the most important features with respect to the whole genomic sequences. We adopted our string selection method on one human virus, human immunodeficiency virus type 1 (HIV-1), and one animal virus, Foot-and-Mouth disease virus (FMDV), to predict their subtypes or recombinant forms. AIDS and FMD are notorious for their perniciousness. Phylogenetic analyses are critical for preparing a strategy to prevent and control those diseases. Our goal is to address the feature composition strings, so as to we can discover more hidden information through them.

The rest of the thesis is organized as follows. We will target on microarray data analysis first and then move on to whole genome virus data analysis. In microarray data analysis, we will first present our Iterative Local Least Imputation method which is for microarray data preprocessing in Chapter 2. Two single ranking gene selection methods, GS1 and GS2, and two clustering based gene selection methods, DCGS-based and ACGS-based gene selection methods, are described in Chapter 3. In Chapter 4, we present the use of gene selection methods to discover the possible blemished entried thus to improve the microarray data quality. In the second part, we first introduce our nucleotide composition string selection in HIV-1 subtyping in Chapter 5. After that, we combine feature selection method in microarray analysis and our string selection method and perform the subtyping for FMDV in Chapter 6.

Chapter 2

Missing Value Estimation

2.1 Introduction and Related Work

[1] As mentioned in Chapter 1, although microarray technology provides us with a good opportunity to understand better the living cell at a genetic level, some technical pitfalls will prevent us from digging up more information from the microarray dataset. One common drawback that might affect the mathematical analysis is the problem of missing values in the datasets obtained from DNA microarray experiments. A number of reasons could lead to missing values, including insufficient resolution, uneven distribution of fluids, and stochastic factors such as image corruption, dust and scratches on the slides and glass flaws. All these could create the artifacts on the microarray chips which result in a certain percentage of expression data corruption [92, 96]. Even with the high-density oligonucleotide arrays such as Affymetrix GeneChip oligonucleotide (Affy) arrays, as high as 20% of expression spots on the arrays could be blemished which may cover hundreds of probes and affect the reading of a considerable percent of gene expression values [92]. Most microarray data analysis applications, such as gene clustering, biomarker identification, sample classification, and genetic and regulatory network prediction, which seek to address biological or medical issues, only accept complete expression values. Therefore, before the data analysis, the gene expression levels have to be preprocessed in order to impute the missing values, as well as correct some portion of the blemished data. One possible solution to this problem is to repeat the experiments [14, 96]. This approach is apparently costly, inefficient, and sometimes infeasible. Therefore, proper mathematical strategies to recover the missing data are desired. An *Iterated Local Least Squares Imputation (ILLSimpute)* method is proposed by us for estimating missing values. The encouraging experimental results on six real microarray datasets show that the ILLSimpute method performs at least as well as, and most of the time much better than, five most recent imputation methods.

In order to further prove our missing value estimation, we also adopt the accuracy of the sample classification through gene selection to measure the quality of the missing value estimation method. That is, missing values are estimated first, followed by gene selection and sample classification.

[1]The ILLSimpute method is published on Journal of Bioinformatics and Computational Biology [18].

We show that the ILLSimpute method can effectively estimate the missing values such that the classification accuracy based on the imputed complete expression level matrices is as high as the classification accuracy based on the original complete expression level matrices.

Several methods have been proposed for effectively imputing the missing values, without doing any extra microarray experiments, through taking advantage of modern mathematical and computational techniques. To name a few, Troyanskaya *et al.* proposed a weighted K-nearest neighbors (KNNimpute) method and a singular value decomposition (SVDimpute) method [96]. Briefly, in the KNNimpute method, for a target gene, k nearest neighboring genes are selected from the entire gene set except those that have missing values at the same positions as the target gene. Later, a weighted linear combination of these nearest neighbors is used to estimate the missing values in the target gene. In the SVDimpute method, a set of mutually orthogonal expression patterns are obtained and linearly combined to approximate the expression of all genes, through the singular value decomposition of the expression matrix $G_{m \times n}$. By selecting k most significant eigengenes, a missing value g_{ij} is estimated by first regressing the i-th gene against these k eigengenes and then using the coefficients of the regression to reconstruct g_{ij} from a linear combination of the k eigengenes [96]. Troyanskaya *et al.* showed that KNNimpute works well on non-time series data and noisy time series data, while SVDimpute performs better on time series data with low noise level [96]. K.-Y. Kim *et al.* proposed a sequential variant of KNNimpute, or sequential K-nearest neighbor (SKNN) imputation [48]. Essentially, SKNN estimates the missing values sequentially from the gene having the least number of missing values, and then uses the imputed values for further imputation. At every iteration, the missing values in the target gene are imputed by the KNNimpute method, using only genes that have no missing value and genes whose missing values have already been imputed. Oba *et al.* proposed a novel missing value estimation method based on Bayesian Principal Component Analysis (BPCA), which estimates a probabilistic model and latent variables within the framework of Bayesian inference [67]. More recently, H. Kim *et al.* applied Local Least Squares (LLSimpute) to estimate missing values [47]. In LLSimpute, a target gene with missing values is again modeled as a linear combination of k nearest neighboring genes, but through local least square optimization.

Besides the above mentioned methods KNNimpute, SKNN, BPCA, and LLSimpute, there are several other recent works: GMCimpute, which is based on Gaussian mixture clustering and model averaging [68] and is compared with KNNimpute; LSimpute, which is based on the least squares principle and utilizes correlations between both genes and arrays [13], and is compared with KNNimpute too; CMVE, which is a collateral missing value estimation [76] and is compared with KNNimpute, BPCA, and LLSimpute; LinCmb, which is a convex combination of several imputation methods [44] and is compared with KNNimpute, SVDimpute, BPCA, and GMCimpute; and LinImp, which fits a gene expression value into a linear model [74] and is compared with KNNimpute.

Among all these imputation methods, there is a common point that when estimating the missing values in a target gene, the expression values of a fixed number of nearest neighboring genes are used. Such a fixed number could be pre-trained or pre-learned using pseudo gene expression data, or could be picked manually. Except SKNN, the above methods adopt the scheme to put row averages, each of which is the average expression value of a gene across all experiments, into the missing value positions as the starting point for imputation. The imputed values then replace the row averages. Note that when estimating a missing entry in the target gene, no genes that also contain a missing value at the same entry can be counted as the nearest neighboring genes to the target gene [96, 67, 48, 47]. Nonetheless, obviously, the imputed values must still depend on the row averages and reporting them as the final estimation might not be completely unbiased.

In this chapter, we propose not to fix the number of nearest neighboring genes used for imputation purpose, but genes that are within a distance threshold (to be determined) to a target gene are all considered *coherent*. We also propose to iterate the imputation for a number of iterations or till the imputation result converges. We combine these two ideas with the LLSimpute method to have *Iterated* LLSimpute, or *ILLSimpute* method. In ILLSimpute, the distance threshold is dataset dependent and is learned using the dataset itself with pseudo missing values. Once the distance threshold is determined, ILLSimpute employs LLSimpute at every iteration. At the first iteration, row averages for missing values are used as the starting point; subsequent iterations use the imputed values from the last iteration as the starting point.

The rest of the chapter is organized as follows: In Section 2.2, we introduce in detail the ILLSimpute method, including how row average is calculated, the definition of distance between two genes, the measure of imputation results, and the LLSimpute method. We explore in Section 2.3 several options in the ILLSimpute method, including the candidate coherent genes for a target gene, the ways to learn the distance threshold, and to impute all missing values in a target gene simultaneously or to impute individual missing values at a time. We report the experimental results in Section 2.4 on the performance of each option in the ILLSimpute method, and the comparison of the default (with the best options) ILLSimpute to five other imputation methods, KNNimpute, SKNN, BPCA, LLSimpute, and LinCmb. Section 2.4.1 also contains the descriptions of the six microarray datasets used in the experiments, and our discussion on the results. We conclude the chapter in Section 2.5.

2.2 Iterated Local Least Squares Imputation — ILLSimpute

One microarray dataset can be represented as an expression matrix $G_{m\times n} = (g_{ij})_{m\times n}$ on m genes and n experiments, where usually $m \gg n$. Entry g_{ij} denotes the expression value of the i-th gene in the j-th experiment. In this section, we present the ILLSimpute method with the default options.

2.2.1 Distance Measure and Coherent Genes

To estimate a missing value, or all the missing values in a target gene, the expression values of a number of nearest neighboring genes, or coherent genes, are used. The determination of these genes relies on the definition of distance between the target gene and a candidate gene. There are several distance measures proposed in the literature, such as Pearson correlation, Euclidean distance, and covariance minimization [96]. We follow the suggestion in [96] to adopt Euclidean distance as it is a sufficiently accurate norm (Euclidean distance is also adopted in KNNimpute and SKNN [48], while LLSimpute adopts Pearson correlation [47] and CMVE adopts covariance minimization [76]). To calculate the distance, we first temporarily fill in the missing value positions in the candidate gene with the row average, which is the average expression value of the gene across all n experiments excluding the missing values. We then ignore in both genes those columns in which the target gene has missing values. This way, we obtain two vectors of expression levels for the target gene and the candidate gene, respectively. The Euclidean distance between these two vectors is taken as the distance between the two genes. Euclidean distance between two vectors $P = (p_1, p_2, \cdots, p_n)$ and $Q = (q_1, q_2, \cdots, q_n)$ is defined as $\sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$. For example, if the target gene is (U, 1.5, U, 2.0, -1.2, U, 2.8) and the candidate gene is (1.6, U, U, -0.4, 2.2, 3.8, U) where U denotes a missing value, then the row average for the candidate gene is $\frac{1}{4}(1.6 - 0.4 + 2.2 + 3.8) = 1.8$. It follows that the two vectors we obtain are (1.5, 2.0, -1.2, 2.8) and (1.8, -0.4, 2.2, 1.8). Therefore, the distance between these two genes is $\sqrt{18.41} = 4.29$.

For every gene, we can calculate its distance to the target gene. Sort the genes in their increasing distance order. The k nearest neighboring genes to the target gene are the first k genes in the list, which are considered coherent genes to the target gene in several imputation methods, including KNNimpute, SKNN, and LLSimpute. In our ILLSimpute method, for a target gene, the average distance is calculated and we will learn a distance ratio δ. The genes with their distances not exceeding δ times the average distance are considered as coherent genes to the target gene.

2.2.2 Local Least Squares Imputation

Once the coherent genes have been determined, the KNNimpute and SKNN methods use a weighted average (a linear combination) of these genes to estimate the missing values in the target gene [96, 48]. In each iteration of our ILLSimpute method, LLSimpute is applied to estimate or re-estimate the missing values. In order to apply LLSimpute, a number of coherent genes must be selected for each target gene. Note that in the original LLSimpute method, these coherent genes are selected as k nearest neighboring genes with k fixed for all target genes. In our case, these coherent genes are selected according to their distances to the target genes. As a result, different target genes might have different numbers of coherent genes. The key step in LLSimpute is using local least squares to determine the coefficients to approximate the target gene as a linear combination of the coherent genes, to be detailed as follows.

Without loss of generality, assume the target gene is gene 1 and it has missing values at the first n' positions. Suppose there are k coherent genes for gene 1 and they are genes $s_1, s_2, \ldots, s_k$. We select the rows $1, s_1, s_2, \ldots, s_k$ from matrix $G_{m\times n}$ to compose the following sub-matrix:

$$\begin{pmatrix} g_1 \\ g_{s_1} \\ \vdots \\ g_{s_k} \end{pmatrix} = \begin{pmatrix} g_{1\times n'} & w^T \\ B_{k\times n'} & A \end{pmatrix} = \begin{pmatrix} g_{1,1} & \cdots & g_{1,n'} & w_1 & \cdots & w_{n-n'} \\ B_{1,1} & \cdots & B_{1,n'} & A_{1,1} & \cdots & A_{1,n-n'} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ B_{k,1} & \cdots & B_{k,n'} & A_{k,1} & \cdots & A_{k,n-n'} \end{pmatrix},$$

where $B_{i,j}$ is the expression value of gene i at position j at which gene 1 has a missing value and $A_{i,j}$ is the expression value of gene i at position j at which gene 1 also has a known expression value of w_j. The k-dimensional coefficient vector x^* is the one that minimizes square $|A^Tx - w|^2 = (A^Tx - w)^T(A^Tx - w)$, that is,

$$x^* = \arg\min_x |A^Tx - w|^2.$$

The target gene is approximated as

$$g_1 \simeq x_1^* g_{s_1} + x_2^* g_{s_2} + \ldots + x_k^* g_{s_k},$$

and the missing values in the target gene are estimated as

$$g_{1j} = x_1^* g_{s_1,j} + x_2^* g_{s_2,j} + \ldots + x_k^* g_{s_k,j}, \quad j = 1, 2, \ldots, n'.$$

2.2.3 Measure of Imputation

When all missing values have been estimated, the quality of the imputation can be measured in several ways. Troanskaya *et al.* [96], and [67, 48, 47], used the *Root Mean Squared Error (RMSE)* between the imputed matrix and the original matrix, divided by the standard deviation of the missing value entries in the complete dataset — the *normalized* RMSE, or NRMSE for short. Jörnsten *et al.* [44] proposed a little change to the NRMSE where the divisor is replaced by the square root of the mean of squared missing value entries in the complete dataset. Besides NRMSE, Kim *et al.* [48] also proposed to use the Pearson correlation coefficients to measure the quality of imputation. With some clustering information known ahead of time, Ouyang *et al.* [68] proposed to use the number of mis-clustered genes to measure the difference, in addition to NRMSE. In this work, we adopt NRMSE defined in [96, 67, 48, 47] to measure the imputation quality, to be detailed as follows. Let $X = \{x_1, x_2, \ldots, x_q\}$ be the set of missing value positions and assume the true expression value at x_i is a_i^* while the estimated value is a_i. Define

$$\mu^2 = \frac{1}{q}\sum_{i=1}^{q}(a_i - a_i^*)^2$$

to be the mean of all squares of errors. Define

$$\overline{a}^* = \frac{1}{q}\sum_{i=1}^{q} a_i^*, \quad \sigma^2 = \frac{1}{q}\sum_{i=1}^{q}(a_i^* - \overline{a}^*)^2$$

to be the mean and variance of all the true expression values, respectively. Then, the NRMSE is defined as

$$\text{NRMSE} = \frac{\mu}{\sigma}.$$

Essentially, the NRMSE indicates how close the estimation values and the true values are. Therefore, the smaller the NRMSE value is, the better quality the imputation has.

2.2.4 Coherent Gene Determination in ILLSimpute

The LLSimpute method selects k nearest neighboring genes for a target gene as its coherent genes. The parameter k is calculated from the dataset prior to the actual imputation. The learning process goes as follows [47]: For every gene, it first fills the missing value positions using the row average. Secondly, it randomly erases a certain number of known expression levels to create the so-called *pseudo* missing values. For every value of k ranging from 1 to the total number of genes in the dataset, the process calls the LLSimpute method once to estimate these pseudo missing values and subsequently calculates the imputation quality, measured by NRMSE. Note that for each of the pseudo missing value, we know the true expression value. The parameter k is then set to the value that achieves the best imputation quality.

In our method ILLSimpute, we do not fix a common number of coherent genes for target genes. Rather than that, the number of coherent genes varies for different target genes. We define the coherent genes for a target gene to be those that are within a distance threshold to the target gene, where the threshold is set as δ times the average distance to the target gene. Here δ is called the distance ratio, which is chosen from a range $[b_1, b_2]$ with an increment ϵ. The impact of setting up a distance threshold rather than a fixed number of coherent genes is that some nearest neighboring genes are already far away from the target gene. The ratio δ is learned using the same scheme as the learning parameter k in the LLSimpute method, namely, for every candidate value of δ, the LLSimpute method is called to estimate the similarly generated pseudo missing values and the corresponding NRMSE is calculated, and the value achieving the best NRMSE is chosen for δ in the subsequent ILLSimpute method to actually impute the missing values. In the default setting of the ILLSimpute method, $b_1 = 0.5$, $b_2 = 1.5$, and $\epsilon = 0.1$.

2.2.5 Iterated LLSimpute — ILLSimpute

Using the learned distance ratio δ (or equivalently, the distance threshold), in the first iteration of our ILLSimpute method, missing value positions are filled with their respective row averages, coherent genes for every target gene are selected, and then the LLSimpute method is executed to estimate the missing values. Afterwards, at each iteration, the ILLSimpute method uses the imputed results from the last iteration to re-select coherent genes for every target gene and executes the LLSimpute method to re-estimate the missing values. Therefore, the only difference between the first iteration and the latter iterations is the use of row averages for selecting coherent genes. The ILLSimpute method terminates after a pre-specified number of iterations or when the re-imputed values in the current iteration have no differences to the imputed values in the preceding iteration, i.e., the imputed values converge. In our implementation, we have decided to set the number of iterations to 5 (Explained later).

2.3 Options in ILLSimpute

In the last section we have introduced the default options in the ILLSimpute method. The method can roughly be partitioned into four modules, which are 1) deciding on candidate coherent genes, 2) learning the distance ratio, 3) executing of LLSimpute method, and 4) iteration. We have examined a number of other options for each of the first three modules and in total 18 versions of the ILLSimpute method. The experimental results showed that the version with the default options performed the best. In the rest of this section, we introduce these other options in detail.

2.3.1 Decisions on Candidate Coherent Genes

In most of previous imputation methods, coherent genes are limited to those that are close to a target gene. There are biological facts showing that some genes could be complementarily expressed [77]. Impacted by these observations, we considered two other categories of potential coherent genes. For ease of exposition, the normally defined nearest neighboring genes are called N-type coherent genes.

Given a target gene, for each other gene, every expression value is multiplied by -1 to produce its ("imaginary") complementary gene. If this complementary gene is within the distance threshold to the target gene, using the Euclidean distance measure, then it is also considered as a coherent gene. This type of ("imaginary") coherent genes are called C-type coherent genes. Note that it could happen that one gene appears as an N-type coherent gene and its complementary gene appears as a C-type coherent gene with respect to the same target gene.

The third type of coherent genes is F-type, which are genes that are far away from the target gene, measured again by the Euclidean distance. The consideration is that these extremely far distance genes might be related to the target gene, similar to the complementary genes. For this purpose, there is another relatively large distance threshold to be set up and one gene that is at least that far away from the target gene is a F-type coherent gene to the target gene. Likewise, this second distance threshold needs to be learned, and is learned in the same way in our implementation.

Since it is obviously inferior to consider only C-type coherent genes or only F-type coherent genes, we have set up three options in choosing coherent genes, one is N-type only, another is N-type and C-type, and the third is N-type and F-type.

2.3.2 Ways of Distance Ratio Learning

In the default option of the ILLSimpute method, the distance ratio δ is chosen from range $[0.5, 1.5]$ with increment $\epsilon = 0.1$. It is set to the value such that the LLSimpute method achieves the best NRMSE on randomly generated pseudo missing values. Note that when the candidate coherent genes can be both N-type and F-type, then there is a second ratio δ' which is chosen from range

$[1.5, 2.5]$ with the same increment ϵ. In the default option, coherent genes are close to individual target genes.

We have also tried to select coherent genes with respect to individual missing values. That is, even for a common target gene, different missing values may have different sets of coherent genes. To do this, the same as before, we fill in pseudo missing value positions their respective row averages. For one pseudo missing value, only its column is excluded in distance calculation. Those genes within the distance threshold are considered as coherent genes for this target missing value.

The third way of learning the distance ratio is almost the same as the second one, except that only those genes with known expression values at the missing value column are considered as candidate genes. The main intention of this way of learning is that row averages are estimations of the pseudo missing values, and it is probably not a good idea to use them to further estimate other pseudo missing values.

For ease of exposition, we call the above ways of distance ratio learning G-way, P-way, and P*-way, to denote the facts that the coherent genes are related to genes, missing value positions, and missing value positions without using row averages for further estimation. Note that those pseudo missing values are randomly generated, and the learned distance ratio(s) might be dependent on those pseudo missing value positions. Therefore, we have tried to repeat the learning several times and taken the average of the resultant distance ratios to be the final ratio(s) in the later actual imputation.

2.3.3 All Versions of ILLSimpute Method

To summarize, we have three ways to select coherent genes, namely N-type only, N-type and C-type, and N-type and F-type; three ways of distance ratio learning, namely G-way, P-way, and P*-way, with or without taking an average over several times, and correspondingly three ways of actual imputation, namely G-way, P-way, and P*-way. Every possible version of the ILLSimpute method corresponds to a path in Figure 2.1 from a box in the leftmost stage to a box in the rightmost stage. There are in total 18 versions and the default version is highlighted in bold.

Note that when the distance ratio has been learned, ILLSimpute can be regarded as a multiple running of LLSimpute. The experiment-wise local least square has a complexity that is about the same as SVD, $O(n^2m)$, where m is the number of genes and n is the number of samples. Therefore, the overall theoretical runtime complexity of ILLSimpute is $O(n^2mm'k)$, where m' denotes the number of target genes (or the number of missing value entries in some versions of ILLSimpute) and k is the number of iterations. Note that the distance ratio learning takes the same amount of time when the number of pseudo missing values is equal to the number of true missing values. We will report in the Experimental Results section the real runtimes of ILLSimpute on six microarray datasets.

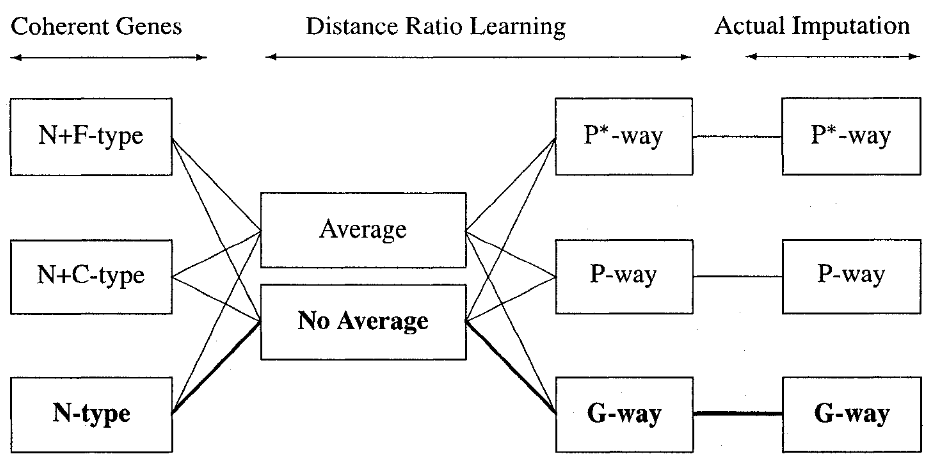


Figure 2.1: 18 versions of ILLSimpute method with the default one highlighted in bold.

2.4 Experimental Results and Discussion

We have set up experiments to examine which option is the best among the available ones for each stage of the ILLSimpute method. In Sections 2.4.2 – 2.4.4, we will report the experimental results on determining the type(s) of coherent genes, choosing the way to learn distance ratio and later actual imputation, and whether or not repeated distance ratio learning is helpful. These results show that the ILLSimpute method with the default options performs the best, in terms of NRMSE. In the subsequent three subsections (Sections 2.4.5 – 2.4.7), we report the experimental results for ILLSimpute method with the default options, on the distance ratio, the number of iterations, and its performance comparison to five other most recently proposed imputation methods, namely, KNNimpute, SKNN, BPCA, LLSimpute, and LinCmb. There are in total six microarray datasets used in the experiments. Section 2.4.1 contains their detailed descriptions. Finally, in Section 2.4.8, we examine the contributions of local least squares over a weighted linear combination and the variable number of coherent genes over a fixed number, both separately and together.

2.4.1 Datasets

We have obtained six microarray datasets for our experiments. The first four datasets are from Spellman *et al.* [84], which were used for identification of cell-cycle regulated genes in yeast *Saccharomyces cerevisiae*. These datasets were obtained from the file "CDCDATA.txt" following link `http://genome-www.stanford.edu/cellcycle/data/rawdata/`. There are three parts in the file: Alpha part, Cdc part, and Elu part. There are 6178 genes in the original file. The first dataset alpha-dataset and the second elu-dataset are the Alpha part and the Elu part in the file, respectively, obtained by removing genes with missing values in either part. Both datasets contain 4304 genes in total, with alpha-dataset consisting of 18 experiments and elu-dataset 14 experiments, respectively.

Again in file "CDCDATA.txt", consider only those C-genes (i.e., YAC, YBC, ..., YPC genes) in the last 14 columns/experiments. Removing genes with missing values gave us cyc-a-dataset that contains 2865 genes in 14 experiments. Alternatively, removing genes as long as they contain a missing value in any column in the original file gave a slightly smaller dataset cyc-b-dataset, which contains 2424 genes in 14 experiments.

The fifth dataset was constructed from a study of response to environmental changes in yeast [34] and can be retrieved through link `http://www-genome.stanford.edu/Mec1/data/DNAcompletedataset/`. The original file contains 6167 genes in 52 experiments. For our purpose, we first removed experiments/columns that have more than 2% missing values, and then removed genes still containing missing values, to obtain env-dataset that contains 5431 genes in 13 experiments.

The above five datasets are time series data with various time setting. The sixth dataset is non-time series and is the cDNA microarray data relevant to human colorectal cancer (CRC) studied in Takemasa *et al.* [94], called ta.crc-dataset, which contains 758 genes in 50 samples/experiments.

Note that the above dataset construction follows some previous studies, and exactly the same alpha-dataset, elu-dataset, and ta.crc-dataset have been used in the studies of BPCA [67] and LL-Simpute [47] methods. From the six microarray datasets we have simulated different missing rates at 1%, 2%, 3%, 4%, 5%, 10%, 15%, and 20% and applied the missing value imputation method to estimate the missing entries.

2.4.2 Comparison of Decisions on Candidate Coherent Genes

Recall that we have tried three different options on candidate coherent genes to cover both similarly expressed genes and potentially complementarity expressed genes. They are N-type only, N-type and C-type, and N-type and F-type. Fixing every option, we have in total 6 versions of ILLSimpute method according to Figure 2.1. On every microarray dataset, each version of the ILLSimpute method was run to obtain its performance measured by NRMSE. The average NRMSE over all 6 versions is taken to measure the quality of this option on the microarray dataset. Table 2.1 includes all these average NRMSE values for three options on six microarray datasets. From the table, we can see that the option "N-type only" has the highest quality (precision in thousandth). A possible reason is that, though there are complementarity expressed genes that could be used for function prediction and other purposes, their expression values might not be very helpful for missing value imputation. These results also demonstrate that similarly expressed genes, measured by the Euclidean distance, do have stronger tie to the target genes than others.

2.4.3 Comparison of Ways of Distance Ratio Learning

Recall that in distance ratio learning we have options to relate coherent genes to individual target genes, individual target missing value positions, or individual target missing value positions but

Option	Dataset					
	alpha	elu	cyc-a	cyc-b	env	ta.crc
N-type only	0.425755	0.359663	0.349580	0.358078	0.629848	0.428770
N+C-type	0.429440	0.360944	0.354819	0.361071	0.633763	0.442413
N+F-type	0.431292	0.360302	0.360616	0.359027	0.628306	0.440117

Table 2.1: Average NRMSE values associated with the three options N-type only, N-type and C-type, and N-type and F-type on candidate coherent genes, overall all six datasets with all 8 missing rates.

excluding genes that have (pseudo) missing values at the same position; they are namely the G-way, the P-way, and the P*-way. We have another global option on whether or not the learning is repeated several times to take the average ratio as the final distance ratio for later actual imputation. We first set up experiments to test the global option. On whether or not the calculation is repeated several times, according to Figure 2.1 there are in total 9 versions of ILLSimpute method. In the repeat option, we have chosen to repeat the calculation 10 times. Again, the average NRMSE value is taken to measure the quality of this option. These average NRMSE values are collected in Table 2.2. From these results, we see that repeating the learning several times does not make a significant difference to the performance of the ILLSimpute method. Therefore, we have decided to set the default option to do the learning only once.

Option	Dataset					
	alpha	elu	cyc-a	cyc-b	env	ta.crc
No Repeat	0.428800	0.360579	0.355083	0.359272	0.630708	0.437444
Repeat 10 Times	0.428858	0.359847	0.354177	0.359512	0.630569	0.436760

Table 2.2: Average NRMSE values of whether or not to repeat the distance ratio learning 10 times, overall all six datasets with all 8 missing rates.

2.4.4 Comparison of Ways of Distance Ratio Learning and Actual Imputation

Since the way distance ratio is learned should be consistent with using it in later actual imputation, we set up experiments to test the quality of each of the three ways: G-way, P-way, and P*-way. Nonetheless, as we have seen that the global option of learning the distance ratio 10 times does not contribute to the performance, we have decided to test only 3 versions of the ILLSimpute method, with the default to learn the distance ratio only once. For each way of distance ratio learning and actual imputation, the average NRMSE value over all three versions of ILLSimpute is taken to measure the quality of the way, which are collected in Table 2.3. From these results, we can see that associating coherent genes with individual target genes is the best way (precision in the thousandths), which is taken as the default.

The above three subsections contain experimental results that support the default options set in

Option	Dataset					
	alpha	elu	cyc-a	cyc-b	env	ta.crc
G-way	0.426085	0.359643	0.353965	0.348193	0.630114	0.433780
P-way	0.429356	0.362107	0.356348	0.362582	0.630895	0.439972
P*-way	0.430961	0.359987	0.354935	0.366771	0.631116	0.438579

Table 2.3: Average NRMSE values of three ways of distance ratio learning and actual imputation, namely G-way, P-way, and P*-way, overall all six datasets with all 8 missing rates.

the ILLSimpute method. In the rest of the section, the ILLSimpute method refers to the default version, unless state explicitly otherwise.

2.4.5 Distance Ratio is Dataset Dependent

In several of the previous imputation methods to select coherent genes, usually a fixed number of them are chosen for every target gene, where that fixed number could be learned or maybe simply manually picked. We implemented the idea to select different numbers of coherent genes for different target genes, as long as the coherent genes are close enough to the target genes. The distance threshold for cutting off dissimilar genes is determined by the distance ratio δ, which is learned using the same dataset. We set up experiments to test if this ratio is dataset dependent or not, as it plays an important role in the actual imputation. Figure 2.2 plots the NRMSE values achieved by ILLSimpute method using different values for the distance ratio on elu-dataset and cyc-b-dataset, both with 10% missing rate (that is, the percentage of missing values). We have tested the distance ratio from 0.5 to 1.5 with an increment of 0.1. It can be seen from Figure 2.2 that the best value for the distance ratio is dataset dependent, as it is 0.6 and 0.9 for elu-dataset and cyc-b-dataset, respectively, though different ratios do not cause big differences on elu-dataset.

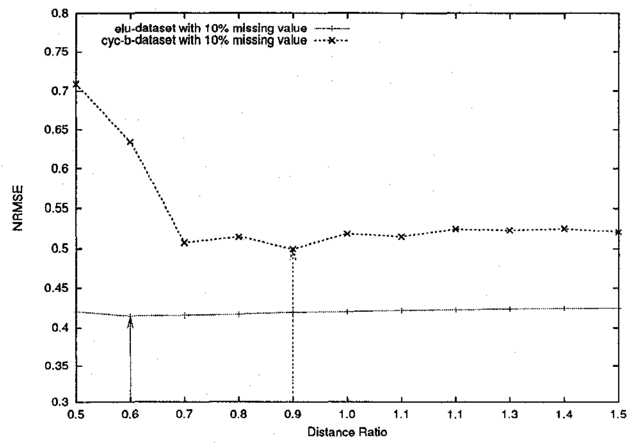


Figure 2.2: The performance of ILLSimpute method on elu-dataset and cyc-b-dataset, both with 10% missing rate, with respect to different distance ratios in $[0.5, 1.5]$.

2.4.6 Determination of the Number of Iterations

The last factor in the ILLSimpute method is to check how many ieterations we will used in this methods. From the massive number of experiments we have done, only a fraction of them converge within 10 iterations, where a convergence happens when the NRMSE difference is less than 10^{-6}. For most of them, ILLSimpute took a large number (in hundreds) of iterations to converge. NRMSE value messure the difference between the estimation values and the values from microarray experiments. Note that, those values from microarray experiments might not accurate, therefore, the convergence points did not always correspond to the best/smallest NRMSE values. Figure 2.3 plots the NRMSE values achieved by the ILLSimpute method after different numbers of iterations, ranging from 1 to 10, on elu-dataset with 10% missing rate and $\delta = 0.6$ and cyc-b-dataset with 10% missing rate and $\delta = 0.9$. On both cases, the best NRMSE values were achieved in 5 iterations. Other experiments also supported choosing to stop in 5 iterations.

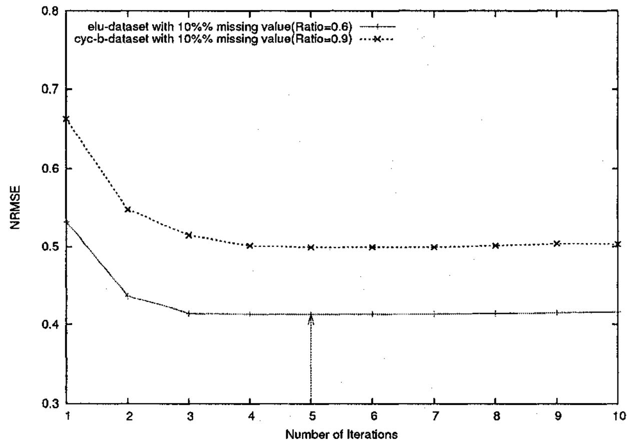


Figure 2.3: The performance of the ILLSimpute method on elu-dataset and cyc-b-dataset with 10% missing rate, after different numbers, 1–10, of iterations.

2.4.7 Performance Comparison of ILLSimpute with Other Methods

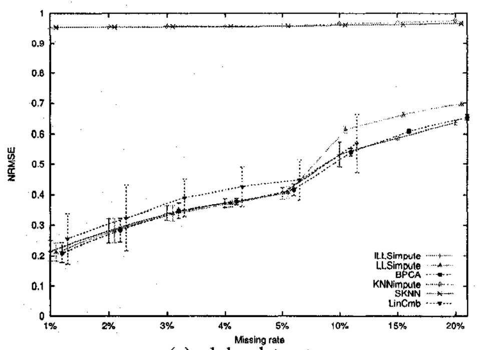
We have compared the performance of the ILLSimpute method on the six microarray datasets to five other most recently proposed missing value imputation methods. ILLSimpute method is set at the default options and terminates in 5 iterations. These five other methods we compared to are KNNimpute, SKNN, BPCA, LLSimpute, and LinCmb. For KNNimpute and SKNN that involve selecting k nearest neighboring genes, we have tested several different values for k and found that $k = 10$ performed the best. The reported results for KNNimpute and SKNN are obtained by setting $k = 10$. For ILLSimpute, we collected the variable numbers of coherent genes that were selected for individual target genes. Surprisingly, the numbers for different target genes within an iteration differ only slightly, but the numbers for common target genes across iterations differ significantly. Also, for different missing rates, the average numbers of coherent genes differ dramatically too. For example, on cyc-b-dataset, the average number is 139 for 1% missing rate, 119 for 10% missing rate, and 197 for 15% missing rate. They differ dramatically too from the number of coherent genes in the LLSimpute method, which we think could be one of the reasons that ILLSimpute performed better than LLSimpute.

Figure 2.4 plots the performances of all six methods, measured by NRMSE, on all six datasets, where the NRMSE is the average over 10 runs. All the plots support the same conclusion that the ILLSimpute method performed at least as well as, and most of the time better than, LLSimpute, and they were significantly better than KNNimpute and SKNN methods. (It is worth pointing out that KNNimpute and SKNN are the most stable methods among them, that is, they have the smallest standard deviations over 10 runs.) BPCA performed equally well to LLSimpute, but mostly inferior to our method ILLSimpute, and the difference became larger when the missing rate increased. For LinCmb, we would not be able to tune it such that all results could be collected within 4 days. Therefore, only those achieved results (using the default setting) are reported, from which we can see that LinCmb performed quite close to BPCA and LLSimpute, but still (much) inferior to our ILLSimpute on the first four datasets. On the fifth env-dataset, LinCmb performed the best among all six, though no result at 20% missing rate. LinCmb completely failed on the sixth ta.crc-dataset, without any result returned (inside LinCmb, GMCimpute did return some partial results).

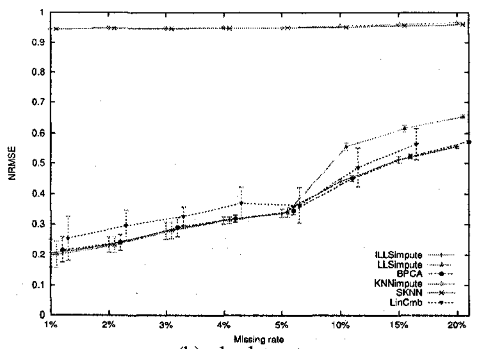
All these imputations were done in MATLAB 7.0 on a cluster of 2.33 GHz CPUs. KNNimpute ran the fastest, and it took only a few seconds each run on each dataset. SKNN, BPCA, LLSimpute, and ILLSimpute took about the same amount of time on each dataset, which is in seconds for some and is in minutes for the others. In general, their runtimes are all acceptable. LinCmb is a convex combination of several imputation methods including GMCimpute. It took already half an hour on several datasets with small missing rates, and the runtime seemed growing exponentially with missing rate, for example some finished run took about 20 hours. Several reported results (each with 10 runs) were achieved in three days, yet still a large portion could not be obtained in a 4-day period. Nonetheless, the readers should be aware that we used the default setting of LinCmb and better tuning of LinCmb might be able to finish those imputations earlier.

2.4.8 Comparison of Fixed and Variable Numbers of Coherent Genes

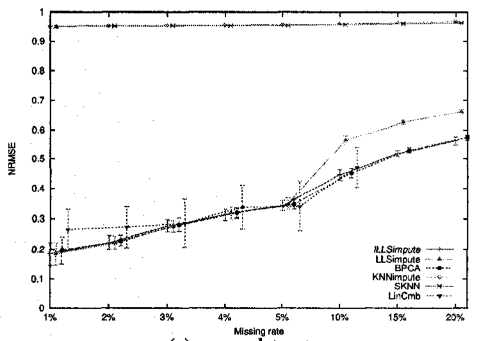
In KNNimpute, a fixed number of coherent genes are used for the purpose of missing value imputation purpose, where the target gene is expressed as a weighted linear combination of its coherent genes. LLSimpute replaces the weighted linear combination by local least square optimization (and so increases the computational complexity by an order of magnitude) and it is shown to perform better in terms of imputation quality NRMSE. We have conducted experiments to validate that, in general, variable numbers of coherent genes for different target genes also outperform a fixed number of coherent genes for all target genes, through replacing the fixed number in KNNimpute by the variable numbers of coherent genes for different target genes determined in the distance ratio



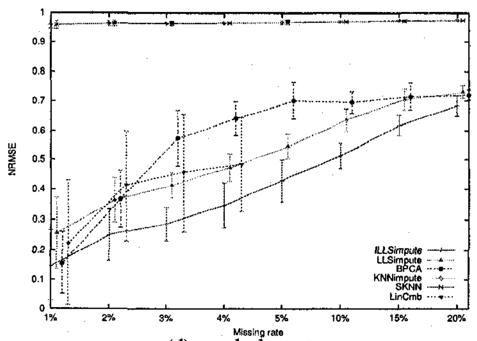
(a) alpha-dataset



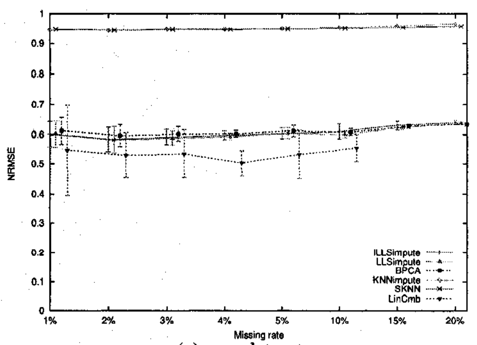
(b) elu-dataset



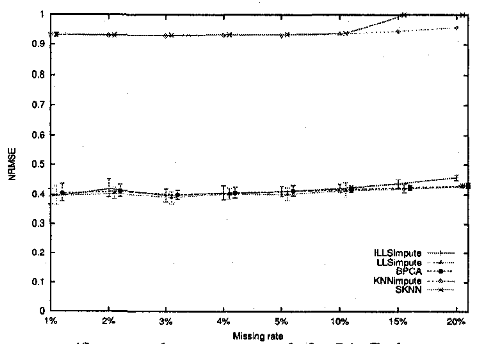
(c) cyc-a-dataset



(d) cyc-b-dataset



(e) env-dataset



(f) ta.crc-dataset: no result for LinCmb

Figure 2.4: Plots of the NRMSE values for ILLSimpute, LLSimpute, BPCA, KNNimpute, SKNN, and LinCmb on six microarray datasets with various missing rates, where the error bars show the standard deviations over 10 runs. LinCmb didn't finish in 4 days on several datasets. These plots show that ILLSimpute performed at least as well as, and most of the time better than, the other five methods. The performance of ILLSimpute becomes significantly better when the missing rate is large.

learning of ILLSimpute. The resultant method is called INNimpute.

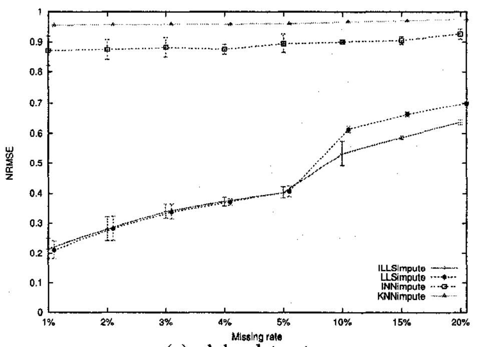
Figure 2.5 plots the performances of all these four imputation methods, measured by NRMSE, on all six datasets. All the plots support the conclusions that, in general, variable numbers of coherent genes is better than a fixed number and local least squares are better than a straightforward weighted linear combination. From the plots, we can see that the ILLSimpute method performed better than the LLSimpute and INNimpute methods on the first four datasets though ILLSimpute and LLSimpute performed equally well on the last two datasets. All these three methods performed significantly better than KNNimpute. Interestingly, these six plots together indicate that when the gain of ILLSimpute over LLSimpute is smaller (on the fifth and the sixth datasets, and some cases of the first three datasets), then the gain of INNimpute over KNNimpute becomes larger. Therefore, we conjecture that in addition to the fact that both variable numbers of coherent genes and local least squares improve the imputation quality, they also complement to each other to pick up the total gain.

2.4.9 Statistical Test on the Performance Differences

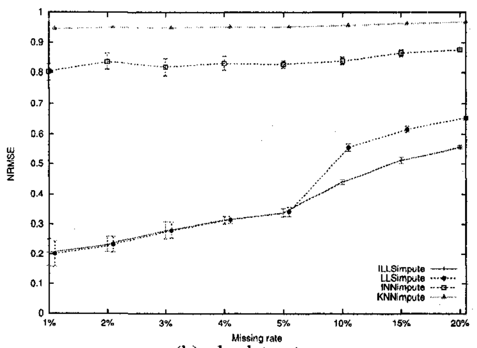
In Subsection 2.4.7, we saw that the collected NRMSE values support the conclusion that the ILLSimpute method performed at least as well as, and most of the time better than, LLSimpute and BPCA (and LinCmb on those datasets LinCmb did finish), and they were significantly better than the KNNimpute and SKNN methods. In this section, we report the statistical tests that were done to evaluate the significance of the performance difference. We adopted *analysis of variance* (ANOVA) for our purpose and used SPSS (`http://spss.com`) to analyze ANOVA. Besides many parameters, ANOVA gives a significance, or p value, of the hypothesis that the two methods under comparison have no difference, at a confidence level, which was set to 95% in our test. Low values of p, less than 0.05, indicate that the hypothesis is false, or the two methods do have difference. Extremely low values of p, less than 0.001, indicate that the difference is significant. ANOVA also gives a 95% confidence interval for the difference between the two NRMSE means associated with the two methods. In general, if the interval contains 0, then the hypothesis is true, or the two methods have no difference. Table 2.4 collects the p values and the 95% confidence intervals associated with elu dataset, for comparing ILLSimpute and each of LLSimpute, BPCA, LinCmb, SKNN, and KNNimpute. Table 2.5 collects the same parameters associated with cyc.b dataset.

2.5 Conclusions

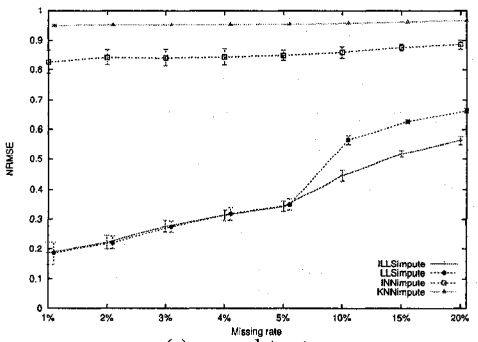
In this chapter, we have introduced our proposed missing value estimation method, a generic iterated version of Local Least Squares Imputation (LLSimpute) method, denoted as ILLSimpute. Within the ILLSimpute method, there are several options on choosing coherent genes for target genes, and selecting the distance ratio learning and actual imputation. Extensive experiments validated that some options appearing to be better do not really contribute to the imputation quality. The ILLSimpute method with default options was compared to five other most recently proposed imputation



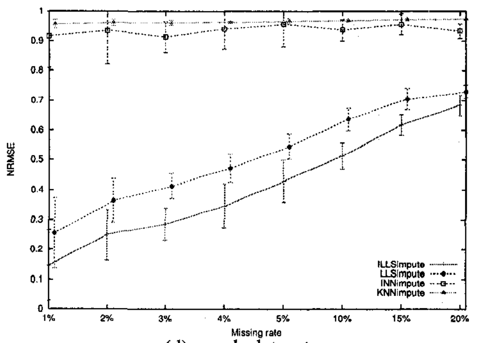
(a) alpha-dataset



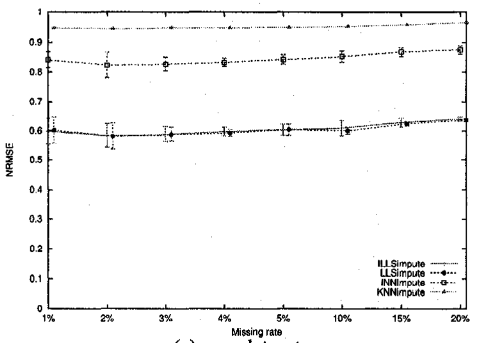
(b) elu-dataset



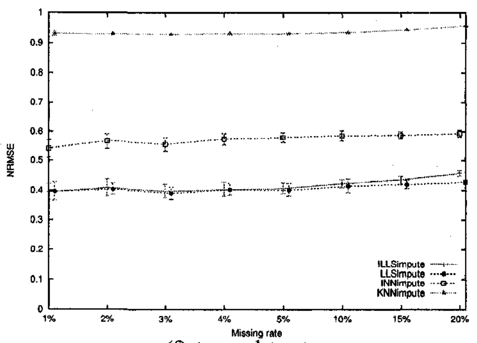
(c) cyc-a-dataset



(d) cyc-b-dataset



(e) env-dataset



(f) ta.crc-dataset

Figure 2.5: Plots of the NRMSE values for ILLSimpute, LLSimpute, INNimpute, and KNNimpute on six microarray datasets with various missing rates, where the error bars show the standard deviations over 10 runs. These plots show that both the use of variable numbers of coherent genes and the iterative imputation improve the imputation quality.

Method	1%			2%			3%			4%		
	p	L	U	p	L	U	p	L	U	p	L	U
LLSimpute	0.999	−0.0538	0.0444	1.000	−0.0330	0.0311	1.000	−0.0272	0.0287	1.000	−0.0250	0.0281
BPCA	0.969	−0.0382	0.0600	0.968	−0.0249	0.0392	0.725	−0.0164	0.0395	0.934	−0.0194	0.0337
LinCmb	0.058	−0.0012	0.0970	0.058	0.0306	0.0947	0.058	0.0202	0.0761	0.000	0.0309	0.0840
SKNN	0.000	0.6911	0.7893	0.000	0.6842	0.7483	0.000	0.6412	0.6971	0.000	0.6108	0.6639
KNNimpute	0.000	0.6917	0.7900	0.000	0.6852	0.7493	0.000	0.6425	0.6983	0.000	0.6124	0.6655

Method	5%			10%			15%			20%		
	p	L	U	p	L	U	p	L	U	p	L	U
LLSimpute	1.000	−0.0275	0.0327	0.000	0.0833	0.1467	0.000	0.0767	0.1284	0.000	0.0933	0.1033
BPCA	0.907	−0.0213	0.0389	0.877	−0.0216	0.0418	0.667	−0.0143	0.0374	0.000	0.0101	0.0201
LinCmb	0.137	−0.0052	0.0549	0.001	0.0154	0.0787	0.000	0.0253	0.0770	–	–	–
SKNN	0.000	0.5829	0.6431	0.000	0.4814	0.5448	0.000	0.4189	0.4706	0.000	0.3997	0.4097
KNNimpute	0.000	0.5843	0.6444	0.000	0.4847	0.5480	0.000	0.4236	0.4753	0.000	0.4061	0.4161

Table 2.4: The p values and the lower bound L and the upper bound U of the 95% confidence intervals in the ANOVA tests of the performance difference between ILLSimpute and each of the other five methods, on elu dataset with 8 different missing rates. These values show that ILLSimpute, LLSimpute, and BPCA (and LinCmb on 1% and 5% missing rates) performed equally well when the missing rate is small, but significantly better when the missing rate is large. All these four methods significantly outperformed the SKNN and KNNimpute methods.

Method	1%			2%			3%			4%		
	p	L	U	p	L	U	p	L	U	p	L	U
LLSimpute	0.165	−0.0284	0.2434	0.043	0.0029	0.2260	0.017	0.0182	0.2370	0.003	0.0354	0.2157
BPCA	1.000	−0.1270	0.1448	0.036	0.0059	0.2291	0.000	0.1780	0.3968	0.000	0.2029	0.3832
LinCmb	0.477	−0.0611	0.2107	0.002	0.0536	0.2768	0.001	0.0615	0.2803	0.001	0.0483	0.2286
SKNN	0.000	0.6755	0.9473	0.000	0.6024	0.8255	0.000	0.5680	0.7868	0.000	0.5270	0.7073
KNNimpute	0.000	0.6756	0.9475	0.000	0.6021	0.8253	0.000	0.5682	0.7870	0.000	0.5270	0.7074

Method	5%			10%			15%			20%		
	p	L	U	p	L	U	p	L	U	p	L	U
LLSimpute	0.000	0.0628	0.1679	0.000	0.0881	0.1577	0.000	0.0536	0.1221	0.000	0.0235	0.0689
BPCA	0.000	0.2182	0.3232	0.000	0.1479	0.2175	0.000	0.0641	0.1326	0.000	0.0140	0.0593
LinCmb	–	–	–	–	–	–	–	–	–	–	–	–
SKNN	0.000	0.4847	0.5897	0.000	0.4215	0.4912	0.000	0.3210	0.3895	0.000	0.2669	0.3122
KNNimpute	0.000	0.4863	0.5913	0.000	0.4223	0.4920	0.000	0.3226	0.3911	0.000	0.2692	0.3145

Table 2.5: The p values and the lower bound L and the upper bound U of the 95% confidence intervals in the ANOVA tests of the performance difference between ILLSimpute and each of the other five methods, on cyc.b dataset with 8 different missing rates. These values show that except when the missing rate is 1%, ILLSimpute performed significantly better then all the other five methods (except LinCmb didn't finish on some datasets).

methods, KNNimpute, SKNN, BPCA, LLSimpute, and LinCmb, on six microarray datasets, and the results demonstrated that our method performed the best in almost all the cases, and most of the time much better than the others. To conclude, we would recommend to use ILLSimpute on time series dataset, especially ones with big missing rates.

Among the six datasets, we have seen that all methods took the longest time on the sixth dataset and especially LinCmb failed. These evidences suggest that the sixth dataset could be the most difficult one, yet it is the only non-time series dataset. ILLSimpute performed the least well on this dataset, which might suggest that non-time series datasets have some unique property that ILLSimpute fails to capture, as well as some of the component methods in LinCmb. This gives another piece of evidence that the ILLSimpute method is good enough to impute the missing value. The quality of the matrix after imputation is good enough for the further microarray analysis.

Chapter 3

Gene Selection

3.1 Introduction

[1] We may regard missing value estimation as data pre-processing. After the pre-processing is finished, we will move on to identify the discriminatory genes. The large amount of microarray data has been employed in wide areas in biological and medical sciences, and interpreted through a biological perspective. One of the most common applications is to compare the gene expression levels in tissues under different conditions, such as wild-type versus mutant, or healthy versus diseased. Generally, the genes measured in microarray experiments belong to two categories. Some genes are usually differentially regulated under various experimental conditions. To be more specific, the expression levels of these genes increase or decrease under certain conditions compared with the normal level. These genes are called discriminatory genes. The expression levels of these genes carry important information related to disease subtype identifying, sample classification, and other purposes [37, 32, 102]. However, these genes only consist of a minority of the genes in microarray. The majority of genes are those whose expression levels are almost constant under different conditions, such as house-keeping genes. They are less important in providing clue for further data analysis and interpretation.

Abstracting useful information from the large volume of data remains one of the most challenging technical problems for the advancement of microarray research. The genes detected by a single microarray experiment usually number thousands to tens of thousands, whereas the number of samples examined is only tens to hundreds. As a result, overfitting is a common problem caused by the high dimensionality of microarray dataset. Both gene selection and dimensionality reduction have been used to reduce the risk of overfitting.

Gene selection is the process of screening for discriminatory genes. A variety of approaches have been proposed for gene selection. For example, Golub *et al.* [37] developed a measure of correlation that emphasizes the "signal-to-noise" ratio in using the gene as a predictor, and selected a number of top ranked genes as discriminatory genes. This ratio captures the basic rule of gene

[1]GS1 and GS2 methods are published in BMC Bioinformatics [102]. ACGS based and DCGS based methods are published in BMC Bioinformatics [16].

selection: that a discriminatory gene must have close expression levels in samples within a class, but significantly different expression levels in samples across different classes. Other approaches that adopt the same principle, with modifications and enhancements, include [9, 24, 102] and many others. Alternatively, Xiong *et al.* [101] selected a subset of genes with a maximum classification accuracy through sequential (floating) forward selections (SFS, SFFS). Guyon *et al.* [39] suggested a gene selection method that uses support vector machines (SVMs) and recursive feature elimination (RFE). Li *et al.* [54] combined a Genetic Algorithm (GA) and k-Nearest Neighbor (KNN) methods to identify feature genes on two classes of dataset. Lee *et al.* [52], Zhou *et al.* [103] and Gawley *et al.* [35] used the Bayesian model to identify significant genes. All these three papers only shot on binary classification. Shevade *et al.* [78] proposed a gene selection method based on sparse logistic regression. Again, the algorithm is only suitable for the two classes of dataset. Diaz-Uriate *et al.* [27] proposed a gene selection method based on random forest, but the feature genes returned by the random forest method are limited. Most of the recently developed methods are only good for binary class data. An efficient and stable gene selection method for multi-class data is needed.

It is of great interest that different gene selection methods report different subsets of genes and they all return high classification accuracies [39, 24, 102]. One explanation is that many genes have similar discrimination power. Including them all would make some portion of the selected gene subset redundant for classification purpose. However, most of the above methods only emphasize on the efficiency of single genes, but overlook the strength of the selected genes as a whole.

Generally, gene selection can be partitioned into two categories, the single ranking gene selection method (univariate method) and the multivariate method [59]. Single gene ranking methods consider the contributions of individual genes to the classification independently. In contrast, multivariate approaches measure the relative contribution of a gene to the classification by considering the effect of other genes at the same time. In the following, we present two single ranking gene selection methods and two multivariate gene selection methods.

3.2 Single Ranking Methods

We propose two novel gene scoring functions $s_1(\cdot)$ and $s_2(\cdot)$ to design two stable gene selection methods GS1 and GS2, respectively, to be detailed next. According to the classification scheme proposed in [97], our proposed gene selection methods are single gene scoring approaches. These two gene scoring functions non-trivially incorporate the means and the variations of the expression values of genes in the samples belonging to a common class, based on a very general assumption that discriminatory genes are those having different means across different classes, small *intra-class variations* and relatively large *inter-class variations*.

Assume that in the training dataset there are in total m genes in the microarray chips, and assume that we have in total n chips/samples that have been grouped into L classes. Let a_{ij} denote the expression value of gene i in sample j. The training dataset can thus be represented as a matrix

$$A_{m\times n} = (a_{ij})_{m\times n}.$$

We will define two gene scoring functions using entry values in matrix $A_{m\times n}$. These two scoring functions might be considered to better use the means and the variations of the gene expression values.

Let n_k denote the number of samples in class C_k, for $k = 1, 2, \ldots, L$ (i.e., $\sum_{k=1}^{L} n_k = n$). Let $\overline{a}_{ik} = \frac{1}{n_k}\sum_{j\in C_k} a_{ij}$, which is the average expression value of gene i in class C_k, for $k = 1, 2, \ldots, L$. The expression vector $(\overline{a}_{1k}, \overline{a}_{2k}, \ldots, \overline{a}_{mk})$ is the *centroid* of class C_k. Correspondingly, the centroid matrix is

$$\overline{A}_{m\times L} = (\overline{a}_{ik})_{m\times L}.$$

The mean of these centroids is $\hat{A} = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_p)$, where $\hat{a}_i = \frac{1}{L}\sum_{k=1}^{L} \overline{a}_{ik}$.

For sample j belonging to class C_k, the difference between the expression value of gene i and the class mean is $x_{ij} = |a_{ij} - \overline{a}_{ik}|$. The matrix

$$X_{m\times n} = (x_{ij})_{m\times n}$$

is the *deviation matrix* of the training dataset. Let $\overline{x}_{ik} = \frac{1}{n_k}\sum_{j\in C_k} x_{ij}$ denote the average deviation for samples in class C_k with respect to the centroid. The centroid deviation matrix is

$$\overline{X}_{m\times L} = (\overline{x}_{ik})_{m\times L}.$$

Intuitively, gene j has a strong discriminating power if the means $\overline{a}_{ik}$, $k = 1, 2, \ldots, L$, differ significantly and $\overline{x}_{ik}$, $k = 1, 2, \ldots, L$, indicating the *intra-class variations*, are all small.

For example, suppose we have a microarray expression matrix $A_{4\times 12}$ as shown, in which 12 samples have been known in 3 classes:

$$A_{4\times 12} = \left(\begin{array}{ccc|cccc|ccccc} 0.65 & 0.85 & 0.9 & 0.9 & 1.1 & 1.5 & 1.3 & 1.2 & 1.5 & 1.7 & 2.0 & 1.6 \\ 0.2 & 1.0 & 1.2 & 0.7 & 1.4 & 1.8 & 0.9 & 1.0 & 1.7 & 2.1 & 2.0 & 1.2 \\ 0.2 & 1.4 & 0.8 & 0.6 & 2.0 & 1.2 & 1.0 & 0.8 & 1.6 & 2.3 & 1.9 & 1.4 \\ 0.7 & 1.0 & 1.3 & 0.5 & 0.7 & 1.6 & 1.2 & 1.5 & 0.2 & 1.8 & 0.3 & 1.2 \end{array}\right).$$

Then the centroid matrix $\overline{A}_{4\times 3}$ is

$$\overline{A}_{4\times 3} = \begin{pmatrix} 0.8 & 1.2 & 1.6 \\ 0.8 & 1.2 & 1.6 \\ 0.8 & 1.2 & 1.6 \\ 1.0 & 1.0 & 1.0 \end{pmatrix},$$

and the mean of the centroids is

$$\hat{A} = (1.2, 1.2, 1.2, 1.0).$$

The deviation matrix $X_{4\times 12}$ is

$$X_{4\times 12} = \left(\begin{array}{ccc|cccc|ccccc} 0.15 & 0.05 & 0.1 & 0.3 & 0.1 & 0.3 & 0.1 & 0.4 & 0.1 & 0.1 & 0.4 & 0.0 \\ 0.6 & 0.2 & 0.4 & 0.5 & 0.2 & 0.6 & 0.3 & 0.6 & 0.1 & 0.5 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.0 & 0.6 & 0.8 & 0.0 & 0.2 & 0.8 & 0.0 & 0.7 & 0.3 & 0.2 \\ 0.3 & 0.0 & 0.3 & 0.5 & 0.3 & 0.6 & 0.2 & 0.5 & 0.8 & 0.8 & 0.7 & 0.2 \end{array}\right),$$

and the intra-class average deviations are

$$\overline{X}_{4\times 3} = \begin{pmatrix} 0.1 & 0.2 & 0.2 \\ 0.4 & 0.4 & 0.4 \\ 0.4 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.6 \end{pmatrix}.$$

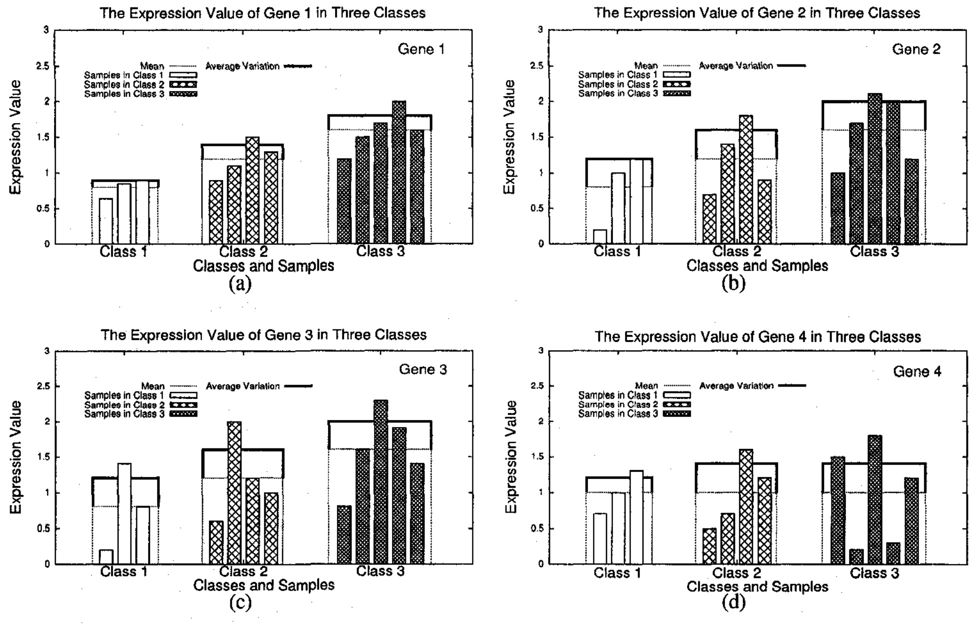


Figure 3.1: An example dataset: the expression values of 4 genes in three classes.

Figures 3.1 illustrates the expression values of these four genes across all 12 samples, with the intra-class means and average deviations also shown. There are three key ideas in our design of gene scoring functions, which will be exemplified through these four genes. First of all, gene 1 has quite different mean expression values across three classes, compared to gene 4 that has the same means. Therefore, gene 1 is intuitively better than gene 4 in terms of discriminating power. Note that the goal of gene selection is to select genes that have significantly different means across different classes. For each gene i, the quantity $\hat{a}_i$ is the mean of all the centroids on gene i and it represents all the samples. $\hat{a}_i$ is stable, that is, it would not change when the samples in one class are duplicated (since the number of classes, L, and all the means, $\overline{a}_{ik}$, for $k = 1, 2, \ldots, L$, do not change). We define the *scatter* of gene i to capture the *inter-class* variations, which takes in $\hat{a}_i$ as a component:

$$scatter(i) = \sqrt{\frac{1}{L}\sum_{k=1}^{L}(\overline{a}_{ik} - \hat{a}_j)^2} + \frac{1}{2}\min_{w\neq v}\left|\overline{a}_{iw} - \overline{a}_{iv}\right|,$$

in which the square root is the standard (estimated) deviation of all the centroids on gene i. If a gene selection method is not affected by the number of samples in each class, we call this method a stable method. It is clearly seen that *scatter*(i) is a stable function. More discriminatory genes are expected to have bigger *scatter* values. In the following, we prove an upper bound and a lower bound for *scatter*(i).

Lemma 3.2.1 *Given n arbitrary non-negative numbers $a_1, a_2, \ldots, a_n$, the inequality $\frac{1}{n}(a_1 + a_2 + \ldots + a_n) \leq \sqrt{\frac{1}{n}(a_1^2 + a_2^2 + \ldots + a_n^2)}$ holds, and the equality holds if and only if $a_1 = a_2 = \ldots = a_n$.*

Lemma 3.2.1 can be proven by a mathematical induction.

Lemma 3.2.2 *Given n arbitrary non-negative numbers sorted in order $a_1 \leq a_2 \leq \ldots \leq a_n$, define $\hat{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$, $\tilde{a} = \frac{1}{2}\min_{1\leq i\leq n-1}(a_{i+1} - a_i)$, and $S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \hat{a})^2}$. Then,*

$$\tilde{a} \leq S \leq (a_n - a_1) - \tilde{a}.$$

PROOF. Note that both S and $\tilde{a}$ are non-negative. Therefore, if $\tilde{a} = 0$, then $S \geq \tilde{a}$ holds trivially. In the other case, we have $a_1 < a_2 < \ldots < a_n$. Assume without loss of generality that the minimum is achieved at $i = k$, that is, $\tilde{a} = \frac{1}{2}(a_{k+1} - a_k)$. If $\hat{a} \in [a_k, a_{k+1}]$, from Lemma 3.2.1, we have

$$\frac{1}{2}\left((a_k - \hat{a})^2 + (a_{k+1} - \hat{a})^2\right) \geq \left(\frac{1}{2}((\hat{a} - a_k) + (a_{k+1} - \hat{a})\right)^2 = \tilde{a}^2.$$

For $i \neq k, k+1$, $(a_i - \hat{a})^2 \geq \tilde{a}^2$. Therefore, $nS^2 = \sum_{i=1}^{n}(a_i - \hat{a})^2 \geq n\tilde{a}^2$. If $\hat{a} \in [a_p, a_{p+1}]$ but $p \neq k$, similarly we will have $\frac{1}{2}\left((a_p - \hat{a})^2 + (a_{p+1} - \hat{a})^2\right) \geq (a_{p+1} - a_p)^2 \geq \tilde{a}^2$ and for $i \neq p, p+1$, $(a_i - \hat{a})^2 \geq \tilde{a}^2$. This proves that $\tilde{a} \leq S$.

Inequality $S + \tilde{a} \leq a_n - a_1$ holds again if $\tilde{a} = 0$, since $(a_i - \hat{a})^2 \leq (a_n - a_1)^2$ for every i. Therefore, we may assume that $a_1 < a_2 < \ldots < a_n$. A similar enlarging process gives $S \leq \max\{a_n - \hat{a}, \hat{a} - a_1\}$. Since

$$a_n - \hat{a} + \tilde{a} \leq a_n - \frac{1}{2}(a_2 + a_1) + \frac{1}{2}(a_2 - a_1) = a_n - a_1$$

and

$$\hat{a} - a_1 + \tilde{a} \leq \frac{1}{2}(a_n + a_{n-1}) - a_1 + \frac{1}{2}(a_n - a_{n-1}) = a_n - a_1,$$

we conclude that $S + \tilde{a} \leq a_n - a_1$. □

According to Lemma 3.2.2, the following theorem on the bounds on *scatter*(j) holds.

Theorem 3.2.3 *For gene j, define $\max(i) = \max_{w\neq v} |\bar{a}_{iw} - \bar{a}_{iv}|$ and $\min(i) = \min_{w\neq v} |\bar{a}_{iw} - \bar{a}_{iv}|$. We have $\min(i) \leq$ scatter$(i) \leq \max(i)$.*

A differentially expressed gene is expected to have not only large inter-class variations, which can be represented by its *scatter*-value, but also small intra-class variations. Secondly, we define a function based on the deviation matrix $X_{m \times m}$ and the centroid deviation matrix $\overline{X}_{m \times L}$:

$$\mu(\overline{X}_{L \times p}, i) = \hat{x}_i = \frac{1}{L}\sum_{k=1}^{L} \overline{x}_{ik} = \frac{1}{L}\sum_{k=1}^{L}(\frac{1}{n_k}\sum_{j \in C_k} x_{ij}),$$

which is stable. Intuitively, the discriminatory genes are expected to have smaller μ-values. In the example dataset, genes 1 and 2 have the same mean expression values across all three classes, that is, they have the equal *scatter* values. Nonetheless, $\mu(\overline{X}_{4\times3}, 1) = 0.167$ and $\mu(\overline{X}_{4\times3}, 2) = 0.4$, and thus gene 1 is better than gene 2 in this sense.

In the same example, we have $\mu(\overline{X}_{4\times3}, 3) = \mu(\overline{X}_{4\times3}, 4) = 0.4$. However, for gene 3, the centroids of three intra-class average deviations are the same, that is, $\overline{x}_{3k} = 0.4$ for $k = 1, 2, 3$; for gene 4, the scenario is totally different, $\overline{x}_{4k} = 0.2, 0.4, 0.6$ for $k = 1, 2, 3$. This raises a question of, based on $\mu(\overline{X}_{m\times L}, i)$, what we can tell about the quality of gene i. The contradictory fact is that gene 3 has a smaller maximum intra-class average deviation and a larger minimum intra-class average deviation. To further differentiate the genes, thirdly, we define function $d_1(i)$:

$$d_1(i) = \sqrt{\frac{1}{L}\sum_{k=1}^{L} \overline{x}_{ik}^2}.$$

From Lemma 3.2.1, $d_1(i) \geq \mu(\overline{X}_{m\times L}, i)$. $d_1(i)$ is also stable, and in the above example we have $d_1(3) < d_1(4)$, which indicates that function $d_1(j)$ could be more sensitive and conservative than function $\mu(\overline{X}_{m\times L}, i)$ on the judgment ability. Another stable function can be defined based on $\mu(\overline{X}_{m\times L}, i)$ is

$$d_2(i) = \sqrt{\frac{1}{L}\sum_{k=1}^{L}\left(\frac{1}{n_k}\sum_{j \in C_k} x_{ij}^2\right)}.$$

Intuitively, $d_2(i)$ includes more details in its calculation than $d_1(i)$ does. In the above example, gene 2 and gene 3 have the same mean expression values across all three classes: $\overline{x}_{i1} = \overline{x}_{i2} = \overline{x}_{i3}$. Therefore, we have $d_1(2) = d_1(3)$ but $d_2(2) < d_2(3)$. Since intuitively gene 2 has a stronger separability than gene 3, $d_2(i)$ could be even more sensitive than $d_1(i)$.

The above two functions $d_1(\cdot)$ and $d_2(\cdot)$ basically consider the means of intra-class variations. The following two functions $\delta_1(i)$ and $\delta_2(i)$ are introduced to capture the variations of intra-class deviations, corresponding to $d_1(\cdot)$ and $d_2(\cdot)$, respectively:

$$\delta_1(i) = \sqrt{\frac{1}{L}\sum_{k=1}^{L}\left(\overline{x}_{ik} - \mu(\overline{X}_{m\times L}, i)\right)^2}, \text{ and } \delta_2(i) = \sqrt{\frac{1}{L}\sum_{k=1}^{L}\frac{1}{n_k}\sum_{j \in C_k}\left(x_{ij} - \mu(\overline{X}_{m\times L}, i)\right)^2}.$$

Theorem 3.2.4 $\delta_1(i) = \sqrt{d_1(i)^2 - \mu(\overline{X}_{m\times L}, i)^2}$ *and* $\delta_2(i) = \sqrt{d_2(i)^2 - \mu(\overline{X}_{m\times L}, i)^2}$.

PROOF. The proof follows by simplifying the definition formulae for $\delta_1(i)$ and $\delta_2(i)$. □

Similar to functions $d_1(i)$ and $d_2(i)$, for an ideal differentially expressed gene i, both $\delta_1(i)$ and $\delta_2(i)$ are expected to have small values. Moreover, similar to the relation between $d_1(i)$ and $d_2(i)$, $\delta_2(i)$ is considered more sensitive than $\delta_1(i)$. We define function $compact_k(i) = d_k(i) + \delta_k(i)$, for $k = 1, 2$, to evaluate the intra-class variations for gene i. And we define the gene scoring function $s_k(i) = compact_k(i)/scatter(i)$ to rank the genes according to their differentiability. Note that a smaller value of $s_k(i)$ indicates a higher differentiability.

We denote the gene selection method using $compact_1(i) = d_1(i) + \delta_1(i)$ as GS1, and the other using $compact_2(i) = d_2(i) + \delta_2(i)$ as GS2. Both GS1 and GS2 are model-free and stable. In each of them, the scores for all genes are calculated and genes are sorted in non-decreasing order of their scores. Since the number of genes, m, is typically much larger than the number of samples, n, the overall running time to compute this order is $O(m \log m)$. In practice, there are several ways to select the informative genes using this order. For example, one may select the top ranked x genes for further analysis, or the top ranked $x\%$ genes, or all the genes with score no larger than some constants, among others.

3.2.1 F-Test Method

The F-test method [32, 28] is also a single gene scoring approach. Besides the notations used in our methods, it uses σ_k^2 to denote the variance of expression value of gene j in the C_k class:

$$\sigma_k^2 = \frac{\sum_{j\in C_k}(a_{ij} - \overline{a}_{ik})^2}{n_k - 1},$$

and $\sigma^2 = \frac{\sum_{k=1}^{L} n_k(n_k-1)\sigma_k^2}{n-L}$ to denote the variance in the whole dataset. Gene j has a score defined to be:

$$F(i) = \frac{\sum_{k=1}^{L} n_k(\overline{a}_{ik} - \hat{a}_j)/(L-1)}{\sigma^2}.$$

3.2.2 Cho's Method

Using the same notations used as in the above, Cho's method [24] defines a weight factor w_j for sample j, which is $\frac{1}{n_k}$ if sample j belongs to class k. Let $W = \sum_{j=1}^{n} w_j$. The weighted *mean*(i) for gene i is defined as

$$mean(i) = \sum_{j=1}^{n} \frac{w_j}{W} x_{ij}.$$

The weighted standard deviation is defined as

$$std(i) = \sqrt{\frac{\sum_{j=1}^{n}(x_{ij} - mean(i))^2}{(n - 1/n)\sum_{j=1}^{n} w_j}}.$$

Then the score of gene j is calculated as

$$C(i) = \frac{mean(i) \times std(i)}{std(\overline{a}_i)},$$

where $std(\overline{a}_i)$ is the standard deviation of centroid expression values $(\overline{a}_{i1}, \overline{a}_{i2}, \ldots, \overline{a}_{iL})$.

3.3 The Datasets

The LEU dataset contains in total 72 samples in two classes, *acute lymphoblastic leukemia* (ALL) and *acute myeloid leukemia* (AML), which contain 47 and 25 samples, respectively. Every sample contains $7,129$ gene expression values. We adopted the pretreatment method proposed in [32] to remove about half the genes and subsequently every sample contains only $3,571$ gene expression values. This dataset was divided into a training dataset with 38 samples, among which 27 and 11 belong to classes ALL and AML, respectively, and a testing dataset with 34 samples, by Golub *et al.* [37].

The SRBCT dataset contains in total 83 samples in four classes, *the Ewing family of tumors* (EWS), *Burkitt lymphoma* (BL), *neuroblastoma* (NB) and *rhabdomyosarcoma* (RMS) [46]. Every sample in this dataset contains only $2,308$ gene expression values. The whole dataset was divided into a training dataset with 63 samples, among which 23, 8, 12, and 12 samples belong to classes EWS, BL, NB and RMS, respectively, and a testing dataset with 20 samples with 6, 6, 3, and 5 in the four classes, respectively [46].

The GLIOMAS dataset [66] contains in total 50 samples in four classes, *cancer glioblastomas* (CG), *non-cancer glioblastomas* (NG), *cancer oligodendrogliomas* (CO) and *non-cancer oligodendrogliomas* (NO), which have $14, 14, 7$ and 15 samples, respectively. Each sample has 12625 genes. We adopted a standard filtering method [66], that is, genes with minimal variations across the samples are removed. For this dataset, intensity thresholds were set at 20 and $16,000$ units. Genes whose expression levels varied < 100 units between samples, or varied < 3 fold between any two samples, were excluded. After preprocessing, we obtained a dataset with 50 samples and 4433 genes.

The LUNG dataset [11] contains in total 203 samples in five classes, *adenocarcinomas*, *squamous cell lung carcinomas*, *pulmonary carcinoids*, *small-cell lung carcinomas* and *normal lung*, which have $139, 21, 20, 6$ and 17 samples, respectively. Each sample has 12600 genes. The genes with standard deviations smaller than 50 expression units were removed and we obtained a dataset with 203 samples and 3312 genes [11].

The CAR dataset [90] contains in total 174 samples in eleven classes, *prostate*, *bladder/ureter*, *breast*, *colorectal*, *gastroesophagus*, *kidney*, *liver*, *ovary*, *pancreas*, *lung adenocarcinomas*, and *lung squamous cell carcinoma*, which have $26, 8, 26, 23, 12, 11, 7, 27, 6, 14$ and 14 samples, respectively. Each sample contains 12533 genes. In our experiment, we preprocessed the dataset as described in [90]. We included only those probe sets whose maximum hybridization intensity (AD) in at least one sample was 200, all AD values $\leq$ 20, including negative AD values, were raised to 20, and whose data was log transformed. After preprocessing, we obtained a dataset with 174 samples and 9182 genes.

The MLL dataset [8] contains in total 72 samples in three classes, *acute lymphoblastic leukemia* (ALL), *acute myeloid leukemia* (AML), and *mixed-lineage leukemia gene* (MLL), which have 24, 28 and 20 samples, respectively. In our experiment, intensity thresholds were set at 100 – 16000 units. Then the relative variation of expressions for each gene was determined by dividing the maximum expression for the gene among all samples (max) by the minimum expression (min), i.e. max/min. We filtered out the genes with $\max / \min \leq 5$ or $(\max - \min) \leq 500$, that is, for max/min fold variation, we excluded genes with less than 5-fold variation and, for (max – min) absolute variation, we excluded genes with less than 500 units absolute. After preprocessing, we obtained a dataset with 72 samples and 8685 genes.

The PROSTATE dataset [81] contains in total 102 samples in two classes *tumor* and *normal*, which have 52 and 50 samples, respectively. The original dataset contains 12600 genes. In our experiment, intensity thresholds were set at 100 – 16000 units, the same as in the MLL dataset. Then we filtered out the genes with $\max / \min \leq 5$ or $(\max - \min) \leq 50$. After preprocessing, we obtained a dataset with 102 samples and 5966 genes.

The DLBCL dataset [80] contains in total 77 samples in two classes, *diffuse large B-cell lymphomas* (DLBCL) and *follicular lymphoma* (FL) which have 58 and 29 samples, respectively. The original dataset contains 7129 genes. We set intensity thresholds at 20 – 16000 units, then we filtered out genes with $\max / \min \leq 3$ or $(\max - \min) \leq 100$. After preprocessing, we obtained a dataset with 77 samples and 6285 genes.

Among the above 8 datasets, the first two LEU and SRBCT have been used frequently for testing gene selection methods and classifiers. For each of them, if we use the ratio of the largest class size divided by the smallest class size to represent the level of unbalance, then the fifth dataset CAR is the most unbalanced. In our experiments, we have run every gene selection method on each of the 8 datasets to rank the genes, and for every $x \leq 80$, the classification accuracies of the classifier built using the top ranked x genes have been collected. We chose to present the accuracies for datasets SRBCT and CAR in details (as plots). For other 6 datasets, we only present two values for x (as tables).

3.4 Performance Measurement

One of the main purposes of gene selection is to identify biomarkers that can effectively predict the classes for samples. To this goal, a number of samples with known class labels are provided, which form the *training* dataset. The classifier built on the selected genes is tested on unlabeled samples and its performance is measured by the *classification accuracy*, which is defined as the number of correctly identified samples divided by the total number of testing samples. The leave-one-out (LOO) cross validation method is a process that uses one sample for testing and all the others for training. Then the process is repeated for every sample in the dataset. Another popular cross validation is ℓ-fold, in which the whole dataset is partitioned into ℓ equal parts and, at one time, one part is used for testing and the other $\ell - 1$ parts for training. In our experiments, we used both cross validation methods, but chose to report only 5-fold average classification accuracy over 20 iterations of random partitions.

We have adopted two ways to build a classifier using the selected genes. One is through Support Vector Machines (SVMs) [39] and the other is through K-Nearest-Neighbor (KNN) search [32]. Essentially, SVMs compute a decision plane to separate a set of objects having different class memberships. There are a number of kernels used in SVMs models for decision plane computing and we chose a linear kernel as described in [39]. A KNN classifier ascertains the class of a query sample by analyzing its K nearest neighbors in the training dataset [32]. We chose the Euclidean distance in our KNN classifier with $K = 5$ and predicted the class by majority vote [32], after testing for several values in the range 4 to 10. The SVM and the KNN we used in MATLAB are from `http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox/`. For ease of presentation, the achieved classifiers are referred to as the SVM-classifier and the KNN-classifier, respectively.

3.5 Experimental Results of Single Ranking Methods

We have applied our gene selection methods GS1 and GS2 based on the gene scoring functions $s_1(\cdot)$ and $s_2(\cdot)$, respectively, to a total of 8 publicly available microarray datasets [85]: the *leukemia* (LEU) dataset [37], the *small round blue cell tumors* (SRBCT) dataset [46], the *malignant gliomas* (GLIOMAS) dataset [66], the *human lung carcinomas* (LUNG) dataset [11], the *human carcinomas* (CAR) dataset [90], the *mixed-lineage leukemia* (MLL) dataset [8], the *prostate* (PROSTATE) dataset [81], and the *diffuse large B-cell lymphoma* (DLBCL) dataset [80], the first two of which have been used for several similar testings of gene selection methods. On each dataset, one portion was used as training dataset for our methods to score the genes, and for each specified number x we reported the classification accuracy of the classifier based on the top ranked x genes on the training dataset. The quality of these top ranked x genes is justified on two aspects: 1) the classification accuracy of the resultant classifier on the testing datasets, and 2) for the first two datasets LEU and SRBCT, the stability when the training datasets were partially changed. All the experiments were conducted in MATLAB (`http://www.mathworks.com/`) environment on a Pentium IV PC with a 2.4GHz processor and a 512MB RAM.

This section reports the classification accuracies for the two classifiers on the testing datasets. Figures 3.2 and 3.3 plot the LOO cross validation accuracies of the SVM-classifier and the KNN-classifier based on four gene selection methods GS1, GS2, Cho's, and F-test, on the SRBCT and CAR datasets, respectively.

Figures 3.4 and 3.5 plot the 5-Fold cross validation accuracies of the SVM-classifier and the KNN-classifier based on four gene selection methods GS1, GS2, Cho's, and F-test, on the SRBCT and CAR datasets, respectively.

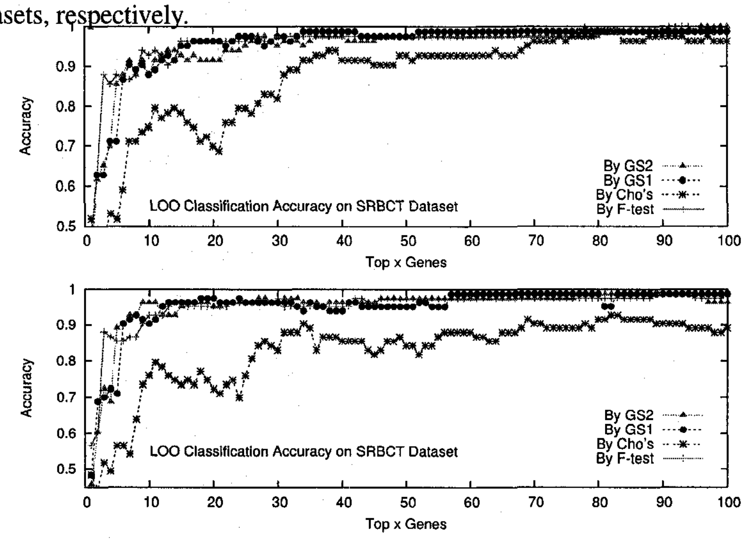
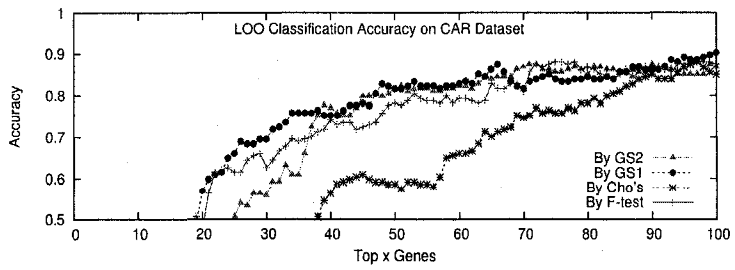


Figure 3.2: The leave-one-out cross validation testing accuracies of the SVM-classifier and KNN-classifier, respectively, on four gene selection methods, GS1, GS2, Cho's, and F-test, on the SRBCT dataset.

Looking at all these testing accuracies, one general conclusion is that our gene selection methods perform at least comparably well to F-test and Cho's, on all 8 datasets using both the SVM-classifier and the KNN-classifier. Typically, our methods outperform significantly the other two methods on datasets SRBCT, GLIOMAS, LUNG, and CAR, which have unbalanced class sizes. Note that, we only show two datasets here.



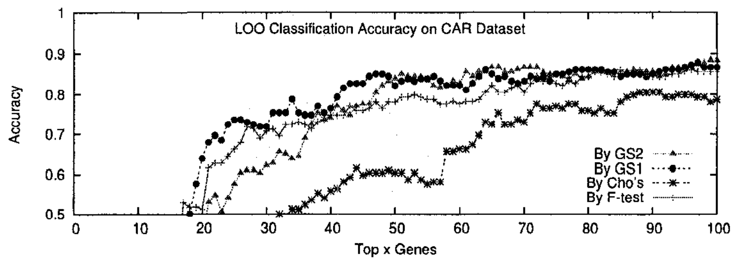


Figure 3.3: The leave-one-out cross validation testing accuracies of the SVM-classifier and KNN-classifier, respectively, on four gene selection methods, GS1, GS2, Cho's, and F-test, on the CAR dataset.

3.6 Stability of the Single Ranking Methods

Given a training dataset, to test the stability of a gene selection method we duplicated all the samples in one class to produce a *duplicated* dataset. Our gene selection methods GS1 and GS2 are shown to be theoretically stable and therefore are not subject to such a test. For each of Cho's and F-test, it was applied on the duplicated datasets to report the same numbers of genes as it was applied to the original training dataset, and then the percentages of the common genes were recorded. Note that the LEU dataset and the SRBCT dataset give 2 and 4 duplicated datasets, respectively. We have done the tests by using only the training datasets and the whole datasets. Table 3.1 collects these percentages.

We have also performed a similar experiment to test the stability when a small portion of samples were removed from the training dataset. For each class in a training dataset, it was divided into three parts of equal size and every time one part was removed from the dataset to obtain a *reduced* dataset. Then again, the percentages of the common selected genes using the original dataset and the reduced datasets were recorded. We tried in total 1000 random divisions and the average of 3000 percentages was taken to be the stability for this class. Table 3.2 collects these stability results for GS1, GS2, Cho's, and F-test. From these results, we can see that GS1, GS2, and F-test had very close stabilities on the reduced datasets, while Cho's had the least over all classes.

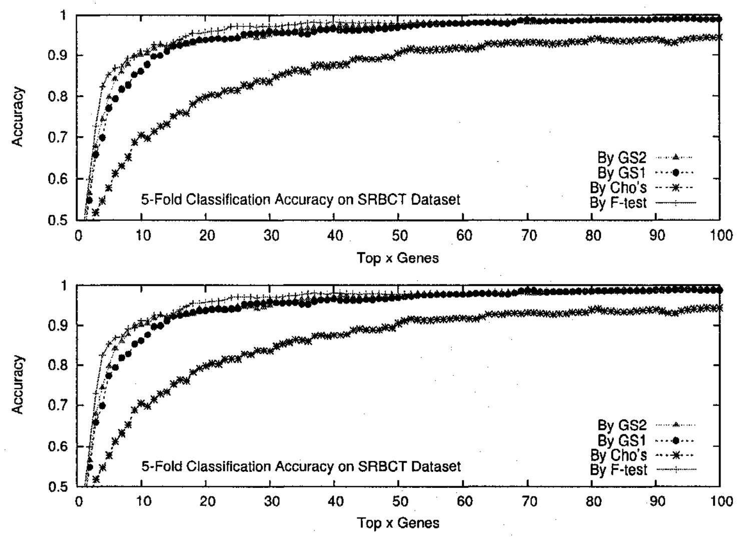


Figure 3.4: The 5-Fold cross validation testing accuracies of the SVM-classifier and KNN-classifier, respectively, on four gene selection methods, GS1, GS2, Cho's, and F-test, on the SRBCT dataset.

Method	x	Whole Dataset					
		SRBCT				LEU	
		EWS	BL	NB	RMS	ALL	AML
Cho's	30	90.0%	93.3%	90.0%	86.7%	96.7%	83.0%
	74	90.5%	89.2%	91.9%	91.9%	93.2%	86.5%
	100	90.0%	94.0%	92.0%	91.0%	92.0%	90.0%
F-test	30	90.7%	85.3%	86.7%	89.3%	83.3%	83.3%
	74	90.7%	85.3%	86.7%	83.3%	86.7%	89.3%
	100	89.0%	87.0%	88.0%	86.0%	92.0%	87.0%

Table 3.1: The percentages of genes that were re-selected by Cho and F-test on duplicated datasets, of the whole LEU and the SRBCT datasets, respectively.

3.7 Combined Clusters and Gene Selection Methods

We propose two cluster based gene selection methods, a direct cluster gene selection (DCGS) method and an additive cluster gene selection (ACGS) method. The cluster gene selection methods are based on the observation that some correlated genes have very similar expression profiles.

The key idea for these two methods is to cluster genes first, and limit the number of genes from the same cluster to be selected. For the DCGS method, we combine gene selection with gene clustering. This method first clusters the genes according to their expression levels across all samples, and assumes genes sharing similar expression profile as similar genes. For the ACGS method, we define similar genes more strictly. The distance between two genes is measured by their discrimination powers. The detailed procedure of this method will be introduced later. Briefly, the ACGS method assigns each gene a discrimination power vector in which each value in this

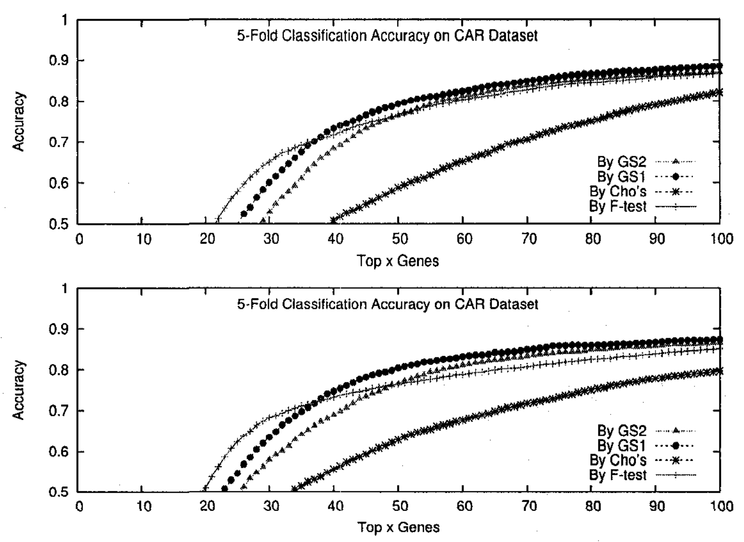


Figure 3.5: The 5-Fold cross validation testing accuracies of the SVM-classifier and KNN-classifier, respectively, on four gene selection methods, GS1, GS2, Cho's, and F-test, on the CAR dataset.

vector stands for the discrimination power between two classes. After that, it clusters genes on genes' discrimination power vectors. In this method, similar genes are defined as genes with similar profiles of discrimination power. After the genes are clustered by the DCGS and the ACGS methods, both methods limit the number of genes from the same cluster to be selected, so as to leave room for dissimilar genes to be selected, as well as some complementary genes that, individually, do not do well at separating the data.

Many existing gene selection methods are based on a gene scoring function that assigns a score for each gene, which approximates the discriminatory strength of the gene. Such gene scoring functions can be the classification accuracy of individual genes [101], or capture the basic rule that discriminatory genes are those being close at expression levels in intra-class samples but being significantly different in inter-class samples [9, 24, 102]. Here we adopt these three gene selection methods, the Cho gene selection [24], F-test gene selection [9, 32], and GS method [2], as our base methods. These methods generally return a number of top ranked genes, and their quality is measured by the classification accuracy of the classifier built on them. It is noticed that some correlated genes have very similar expression profiles. Consequently, they must have similar discrimination power in terms of classification, and once one is top ranked, the others are also likely to be top ranked. However, using them all in classification is redundant in that their discrimination power overlaps. Furthermore, when the top ranked genes are selected, they occupy the spaces for other useful genes to be considered, thus eventually preventing a more efficient classifier.

Based on the above ideas, we propose to do gene clustering before gene selection. Assume the training dataset consists of m genes and n samples. We apply a k-means clustering algorithm to group genes into k clusters by their expression profile or discrimination strength vector, corresponding to two methods (DCGS and ACGS, to be defined below), respectively. Essentially,

[2]Refer to GS1 method in the previous section.

x	Method	Whole Dataset					
		SRBCT				LEU	
		EWS	BL	NB	RMS	ALL	AML
30	GS2	87.5%	81.3%	85.0%	83.8%	87.4%	84.9%
	GS1	82.9%	73.9%	80.4%	81.8%	85.5%	80.4%
	Cho's	83.2%	79.9%	85.5%	83.4%	75.0%	79.0%
	F-test	92.2%	92.1%	90.8%	93.3%	84.4%	80.1%
74	GS2	85.5%	82.6%	86.4%	83.9%	84.5%	80.8%
	GS1	84.6%	80.2%	84.3%	85.2%	83.0%	80.9%
	Cho's	87.5%	86.9%	88.3%	85.0%	75.8%	77.0%
	F-test	87.6%	92.2%	89.3%	88.0%	83.6%	80.7%
100	GS2	86.6%	83.9%	87.6%	86.3%	83.7%	80.8%
	GS1	84.9%	79.2%	84.1%	82.9%	83.6%	80.9%
	Cho's	88.7%	84.5%	89.5%	86.0%	77.4%	75.8%
	F-test	89.3%	92.0%	89.5%	89.2%	83.0%	83.9%

Table 3.2: The percentages of genes that were re-selected by Cho and F-test on reduced datasets, of the whole LEU and the SRBCT datasets, respectively.

k-means is a centroid-based clustering algorithm that partitions the genes into k clusters based on their pairwise distance, to ensure that intra-cluster similarity is high and inter-cluster similarity is low. Both Euclidean distance (`http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/software.htm`) and Pearson correlation coefficient (`http://rana.lbl.gov/EisenSoftware.htm`) are adopted in k-means algorithm. At the same time, a gene scoring function is used to order the genes within the clusters.

A direct cluster gene selection method (DCGS) and an additive cluster gene selection method (ACGS) are adopted in this chapter. For the DCGS methods, the distance between two genes is measured by Euclidean distance (Pearson correlation coefficient) of their expression values directly. After that, the k-means algorithm is applied. In the meanwhile, a gene scoring function is used to order the genes within the clusters.

For gene clustering using the ACGS method, the distance measurement between two genes is more complicated. Firstly, we assign a vector called the *discrimination power vector*, to each gene. The value in the vector represents the discrimination power of that gene between a pair of classes.

The discrimination power vector of gene j is denoted as vector $(|\overline{a}_{k_1 j} - \overline{a}_{k_2 j}|)_{1\times\frac{L\times(L-1)}{2}}$ where and $1 \leq k_1 < k_2 \leq L$. Correspondingly, the discrimination power matrix for all the genes can be represented as

$$D_{m\times\frac{L\times(L-1)}{2}} = (|\overline{a}_{ik_1} - \overline{a}_{ik_2}|)_{m\times\frac{L\times(L-1)}{2}}$$

where $1 \leq k_1 < k_2 \leq L$. Note that, each column in matrix D can be identified as the discrimination power of the genes on a pair of classes.

For example, suppose we have a microarray expression matrix A_3 as shown, in which 9 samples have been known in 3 classes, C_1, C_2, C_3 in which, $\{S_1, S_2\} \in C_1$, $\{S_3, S_4, S_5\} \in C_2$ and $\{S_6, S_7, S_8, S_9\} \in C_3$

$$A_{3\times 9} = \left(\begin{array}{c|cc|ccc|cccc} & S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9 \\ \hline G_1 & 1.3 & 1.7 & 3.0 & 3.1 & 2.9 & 1.6 & 1.7 & 1.6 & 1.5 \\ G_2 & 3.2 & 3.0 & 1.5 & 1.2 & 1.8 & 2.9 & 3.1 & 2.8 & 3.2 \\ G_3 & 0.8 & 0.6 & 2.1 & 2.2 & 2.3 & 0.9 & 0.7 & 0.5 & 0.7 \\ G_4 & 0.7 & 0.9 & 0.8 & 0.9 & 1.0 & 2.3 & 2.4 & 2.6 & 2.7 \end{array}\right).$$

The centroid matrix $\overline{A}_3$ is

$$\overline{A}_{3\times 3} = \left(\begin{array}{c|ccc} & C_1 & C_2 & C_3 \\ \hline G_1 & 1.5 & 3.0 & 1.6 \\ G_2 & 3.1 & 1.5 & 3.0 \\ G_3 & 0.7 & 2.2 & 0.7 \\ G_3 & 0.8 & 0.9 & 2.5 \end{array}\right),$$

and the discrimination power vector is

$$\begin{aligned} v_1 &= \ <1.5, 0.1, 1.4>, \\ v_2 &= \ <1.6, 0.1, 1.5>, \\ v_3 &= \ <1.5, 0.0, 1.5>, \\ v_4 &= \ <0.1, 1.7, 1.6>. \end{aligned}$$

Therefore, the pairwise Euclidean distances are $d(1,2) = 0.141, d(1,3) = 0.141, d(1,4) = 2.135, d(2,3) = 0.141, d(2,4) = 2.195, d(3,4) = 2.205$. The first three genes have similar discriminatory strength, while the G_4 has different ones.

After that, we apply the k-means clustering algorithm [12] to group these m genes into k clusters based on their discrimination power vectors $D_{m\times\frac{L\times(L-1)}{2}}$. Note that for genes selected by the ACGS method, the genes in the same group should have similar discrimination power. At the same time, a gene scoring function is used to order the gene within clusters.

After the clustering is finished by the DCGS or the ACGS method, we pick in total L genes such that at most T genes from each cluster are selected, that is, when there are already T genes from a cluster, other genes belonging to the same cluster are simply skipped.

By doing this, we avoid enrolling genes with similar expression pattern or discrimination power into the gene subset even all of them rank high individually. Genes in the same cluster are usually similar genes. We assume these genes, either sharing the expression pattern or having the similar ability to discriminate certain genes, function identically or similarly in classification. Limiting the number of genes in each cluster facilitates their complementary genes that are more characteristic in other types of samples to be selected. Our methods attempt to maximize the subsets where genes are selected from, such that we can have genes to distinguish each pair of classes.

Depending on the scoring function used, which can be Cho, F-test, or GS, we combine the scoring function name and clustering method name to generate gene selection method names. The resulting names are DCGS-Cho, DCGS-F-test, DCGS-GS, ACGS-Cho, ACGS-F-test and ACGS-GS, respectively. Subsequently, these L selected genes are fed to a KNN-classifier [102] and an SVM-classifier with a linear kernel (`http://theoval.sys.uea.ac.uk/~gcc/svm/`

`toolbox/`), to construct the class predictor. For example, DCGS-GS-SVM refers to the SVM predictor built on genes selected by the DCGS-GS method.

3.8 Experimental Results of Combined Methods

3.8.1 Cross Validation Classification Accuracies

We report the experimental results on choosing the value $k = 100$ in the k-means clustering algorithm. For another parameter T, which is the maximum number of genes from the same cluster, we choose $T = 1$. The reason we choose $k = 100$ and $T = 1$ is that it is the best setting that often achieves the highest classification accuracies in our experiments. Figure 3.6 and Figure 3.7 show 5-fold cross validation classification accuracies over 20 partitions for the Cho, F-test, GS, DCGS-Cho, DCGS-F-test, DCGS-GS, ACGS-Cho, ACGS-F-test and ACGS-GS gene selection methods on the CAR and LUNG, respectively, where KNN-classifier and SVM-classifier are plotted separately.

We also collected the classification accuracies by using sequential forward search (SFS) method [101] and sequential forward floating search (SFFS) method [101]. The performances of those two methods are slightly better than Cho, F-test, GS methods, but much worse than ACGS-based and DCGS-based methods [79].

The classification accuracies on all the three datasets show that the ACGS based methods and DCGS based methods are much better than their non-ACGS-DCGS counterparts. Moreover, ACGS based methods outperform the DCGS based methods a little bit on the CAR and the LUNG datasets.

3.8.2 Standard Deviations

Figure 3.8 shows the standard deviations over 100 5-fold classification accuracies of three types of gene selection methods, combined with KNN-classifier and SVM-classifier, on the CAR and LUNG datasets, respectively. ACGS-KNN is denoted as the average standard deviation of the ACGS-Cho, ACGS-F-test and ACGS-GS methods combined with the KNN classifier. In the same way, we define the DCGS-SVM, DCGS-KNN, DCGS-SVM, non-ACGS-DCGS-KNN and non-ACGS-DCGS-SVM where non-ACGS-DCGS is denoted as three base methods, the Cho, F-test and GS method. The results show that the standard deviations of accuracies of the ADGS-based and DCGS-based methods do not have significant difference within a dataset, and both of them are better than non-ACGS-DCGS methods.

3.9 Discussion of Combined Methods

3.9.1 Number of Clusters and Number of Genes Per Cluster

In the gene clustering method, the k-means algorithm requires a manual input k that defines the number of the clusters as input. This value of k would affect the sizes of resultant clusters. Another manual setting is T, which restricts the maximum number of genes from the same clusters. Both of them are important parameters in our ACGS-based and DCGS-based gene selection methods. Nevertheless, we suppose we do not have any previous knowledge about the datasets. Figure 3.9 shows that the classification accuracy can reach high when $T = 1$. And it seems from our experiments that the k number did not affect the accuracy significantly. The accuracies are almost the same on different k values. Based on the experiments, we decided to choose $k = 100$ and $T = 1$ as our default setting. We report the classification accuracy on the CAR dataset by choosing different combinations of k values, where $k = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150$, and T value, where $T = 1, 2, 3, 4, 5$. For each k, its quality is measured by the average 5-fold classification accuracy over 5 values of T, three gene selection methods and two classifiers, that is, we have in total $100 \times 5 \times 3 \times 2 = 3000$ classification accuracies. In the same way, for each T value, its quality is measured by the average 5-fold classification accuracies over 15 values of k, that is, a total of $100 \times 15 \times 3 \times 2 = 9000$ classification accuracies. All these average classification accuracies are plotted in Figure 3.9. The same setting applies for the DCGS-based gene selection methods. Therefore, for both ACGS and DCGS based gene selection method, we adopted $T = 1$ and $k = 100$ as our default setting.

3.9.2 Distance Measure in k-Means Clustering

In gene expression microarray data analysis, Euclidean distance and Pearson correlation coefficient are two most commonly used similarity measurements between discrimination power vectors. Both are tested in the k-means clustering algorithm on the LUNG and CAR datasets. The results in Figure 3.10 show that one similarity measure is not dominantly better than the other using either ACGS or DCGS gene selection methods. We choose Euclidean distance as our default setting.

3.9.3 Statistical Test on the Performance Differences

All the details ANOVA for all non-ACGS-DCGS gene selection combined with a classifier and their corresponding ACGS-based methods were collected. Table 3.3 shows the same parameters associated with the CAR dataset. The results show that ACGS-based methods and DCGS-based methods are significantly better than their corresponding non-ACGS-DCGS gene selection methods, and ACGS-based methods and DCGS-based methods did not show significant difference.

3.9.4 ℓ-fold Classification Accuracy Determination

In general, ℓ-fold cross validation and Leave-one-out cross validation are applied for the classification accuracies comparison. Actually Leave-one-out (LOO) cross validation is a special case of ℓ-fold cross validation when ℓ equals to the number of samples. We test out the results on both LOO cross validation and ℓ-fold cross validation when $\ell = 3, 5, 8, 10$. The first three figures in Figure

ACGS vs. DCGS	# Genes=20			# Genes=40			# Genes=60			# Genes=80		
	p	L	U	p	L	U	p	L	U	p	L	U
Cho-KNN	0.204	0.0066	0.3815	0.003	0.0096	0.0529	0.007	0.0065	0.0457	0.034	0.0014	0.0418
Cho-SVM	0.064	0.0012	0.4951	0.000	0.0194	0.0614	0.002	0.0101	0.0525	0.197	-0.0053	0.0318
F-test-KNN	0.678	-0.1840	0.0368	0.026	0.0026	0.0484	0.001	0.0129	0.0527	0.012	0.0047	0.0430
F-test-SVM	0.395	-0.1354	0.0439	0.004	0.0087	0.0509	0.000	0.0141	0.0525	0.002	0.0091	0.0443
GS-KNN	0.110	-0.0044	0.0530	0.649	-0.0169	0.0353	0.185	-0.0057	0.0363	0.037	0.0009	0.0364
GS-SVM	0.005	0.0100	0.0638	0.255	-0.0088	0.0416	0.002	0.0105	0.0522	0.014	0.0036	0.0366

ACGS vs. Original	# Genes=20			# Genes=40			# Genes=60			# Genes=80		
	p	L	U	p	L	U	p	L	U	p	L	U
Cho-KNN	0.000	0.3806	0.4253	0.000	0.2978	0.3411	0.000	0.1911	0.2303	0.000	0.1262	0.1666
Cho-SVM	0.000	0.3584	0.4091	0.000	0.3486	0.3906	0.000	0.2315	0.2739	0.000	0.1402	0.1773
F-test-KNN	0.000	0.2824	0.3377	0.000	0.1203	0.1660	0.000	0.0955	0.1353	0.000	0.0542	0.0925
F-test-SVM	0.000	0.3089	0.3663	0.000	0.1372	0.1794	0.000	0.0872	0.1256	0.000	0.0610	0.0963
GS-KNN	0.000	0.3464	0.4038	0.000	0.1048	0.1571	0.000	0.0436	0.0856	0.000	0.0020	0.0374
GS-SVM	0.000	0.3766	0.4304	0.000	0.1006	0.1510	0.000	0.0598	0.1015	0.000	0.0267	0.0597

DCGS vs. Original	# Genes=20			# Genes=40			# Genes=60			# Genes=80		
	p	L	U	p	L	U	p	L	U	p	L	U
Cho-KNN	0.000	0.3648	0.4095	0.000	0.2666	0.3099	0.000	0.1650	0.2042	0.000	0.1046	0.1450
Cho-SVM	0.000	0.3343	0.3850	0.000	0.3081	0.3501	0.000	0.2001	0.2425	0.000	0.1270	0.1640
F-test-KNN	0.000	0.2732	0.3285	0.000	0.0948	0.1405	0.000	0.0627	0.1025	0.000	0.0303	0.0687
F-test-SVM	0.000	0.2937	0.3511	0.000	0.1074	0.1495	0.000	0.0539	0.0923	0.000	0.0344	0.0696
GS-KNN	0.000	0.3221	0.3795	0.000	0.0956	0.1478	0.000	0.0282	0.0702	0.000	0.0228	0.0540
GS-SVM	0.000	0.3397	0.3935	0.000	0.0843	0.1346	0.000	0.0285	0.0702	0.004	0.0066	0.0396

Table 3.3: The p values and the confidence intervals in ANOVA tests of performance among ACGS-based methods, DCGS-based methods and their corresponding non-ACGS-DCGS based methods on CAR dataset by 5-Fold cross validation. Note that, we have 100 classification accuracies when the number of genes z for gene selection is fixed. In this table, we choose $z = 20, 40, 60, 80$.

3.11 show that when ℓ becomes larger, the classification accuracy becomes better. The reason for these results are, when the amount of training samples increased, more about the testing dataset can be captured. The last figure in Figure 3.11 is the average classification accuracy among the DCGS-based, ACGS-based and non-ACGS-DCGS-based methods. Each plot is the average classification accuracy over three gene selection methods and two classifiers, that is, a total of $3 \times 2 \times 100$ classification accuracies. The result show that, the ACGS-based method and the DCGS-based method are much better than the non-ACGS-DCGS method no matter which value of ℓ we choose.

3.9.5 Covariance

As we mentioned before, the ACGS-based method and the DCGS-based methods are designed to avoid the situation where too many similar genes are selected. Covariance is a measurement of the strength of the correlation between two or more sets of random variables, that is, the larger value between the two sets of variables indicate the stronger the strength of relationship. Table 3.4 shows the average absolute covariance value among all pairs of subset genes selected by different gene selection methods. Firstly, we ran the 5-fold cross validation 20 times, that is, there were in total 20×5 subsets of genes being selected. After that, we calculated the frequency of a gene occurring in these 100 subsets. The top 80 genes with most frequent value were selected and the covariance between each pair of these 80 genes were obtained. The average absolute covariance values are listed in Table 3.4. The non-ACGS-DCGS-based methods provide the largest absolute covariance, that is, the genes selected by the non-ACGS-DCGS-based methods have the strongest relationship or vary together in the same direction or opposite direction from their means. The ACGS-based methods obtain the smallest covariance value, the expression value of these genes vary together in different directions.

Methods	Cho		F-test		GS	
	Covariancd	Standard Deviation	Covariancd	Standard Deviation	Covariancd	Standard Deviation
DCGS-based	0.4059	0.0817	0.4421	0.0833	0.4351	0.0844
ACGS-based	0.4340	0.0777	0.5060	0.0797	0.4380	0.0702
non-ACGS-DCGS-based	0.5005	0.1202	0.6562	0.1643	0.5436	0.1442

Table 3.4: Covariance and standard deviation between the DCGS-base, ACGS-based and non-ACGS-DCGS-based methods.

3.10 A Case Study: The CAR dataset

The DCGS based and ACGS based gene selection methods are generally superior to the corresponding non-ACGS-DCGS ones. To investigate whether the DCGS and ACGS based methods have chosen the biologically relevant genes as the discriminators, the top 80 genes were analyzed further for all nine methods ACGS-Cho, DCGS-Cho, Cho, ACGS-F-test, DCGS-F-test, F-test, ACGS-GS, DCGS-GS, and GS. Compared with the tumor-specific 216 genes published in the paper of the CAR dataset [90], the top 80 genes selected by the three non-ACGS-DCGS based methods share 39, 41 and 43 genes, respectively.

In contrast, the three corresponding DCGS methods selected only 17, 17, 20 genes, and three corresponding ACGS methods selected only 18, 16, 17, genes, respectively.

The first impression is that this reduced percentage of discriminatory genes does not make sense, but further investigation showed that the ACGS methods and DCGS method identified many more cancer related genes, to be explained in the following.

Overall, there are only 3 common genes selected by all the nine methods, which are KLK3, ELA3A, and GATA3. KLK3 and ELA3A are signature genes for prostate and pancreas tumors, respectively [90]. All these results clearly demonstrate that different gene selection algorithms can select different sets of genes for disease classification. The experiments also show that, in principle, there exist genes having equal discrimination power, as different sets of genes can reach compatible classification accuracies on the same dataset.

It is particularly interesting to find out that the gene sets selected by all three DCGS based methods share less with the published tumor-specific genes, but they all achieved significantly better classification accuracies. The comparison of the top 80 genes between these three sets revealed a subset of 29 genes, which were picked rarely by the three non-ACGS-DCGS based methods (3, 9, 10, by Cho, F-test and GS, respectively). An immediate question is whether this subset plays an important role for achieving the better classification accuracies for the DCGS based methods. To answer this question, the probe sets of these 29 genes were examined individually to investigate their biological functions related to the intrinsic molecular characteristics of cancer cells. These 29 probe subsets actually represent 28 genes, due to the redundancy of two probe subsets targeting PRCKI. The collected facts clearly show that the majority of this set of 28 genes are related to the human tumorigenesis.

For the ACGS based method, common genes selected by the ACGS based methods are 40, including 14 genes which are the common feature genes selection by the DCGS based methods. For the other 26 genes, we found 19 feature genes which are associated with cancer. That is, there are in total 31 feature genes over 40 which we found related with cancer. Again, among these 40 genes, only 8, 13 and 14 genes were picked by the Cho, F-test and GS methods, respectively.

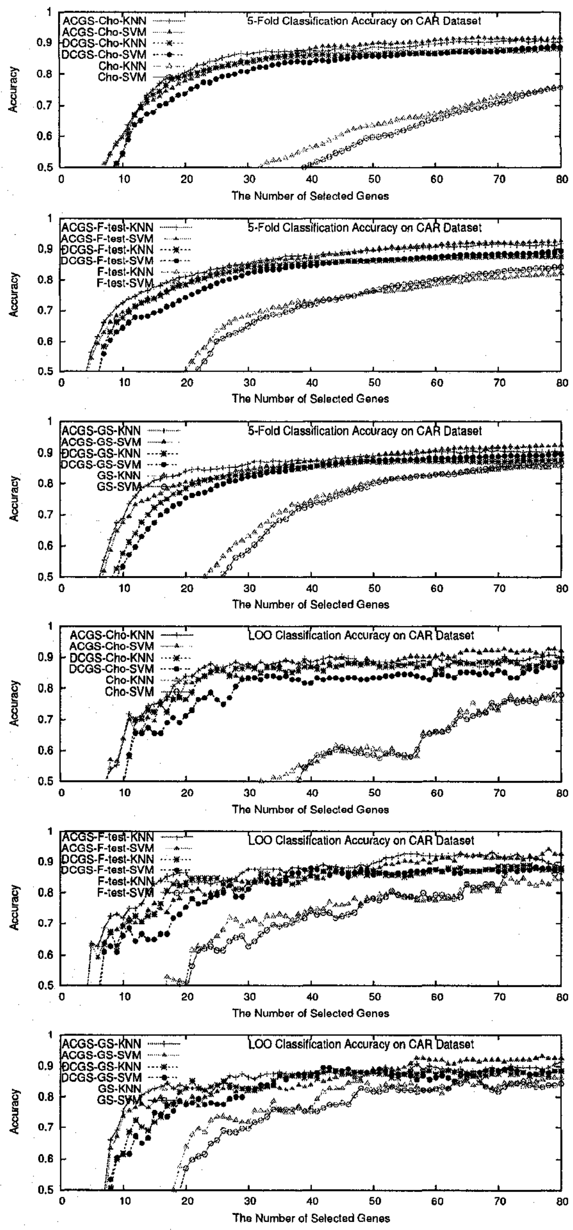


Figure 3.6: 5-fold cross validation accuracies and LOO cross validation accuracies of all 9 methods, combined with KNN-classifier and SVM-classifier, on the CAR dataset.

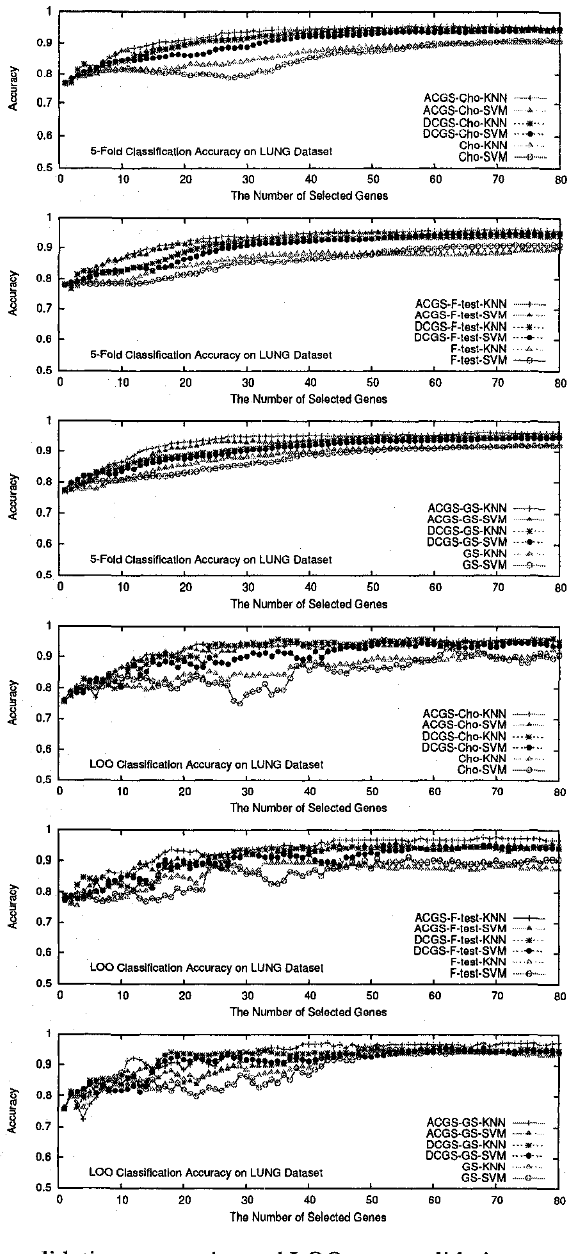


Figure 3.7: 5-fold cross validation accuracies and LOO cross validation accuracies of all 9 methods, combined with KNN-classifier and SVM-classifier, on the LUNG dataset.

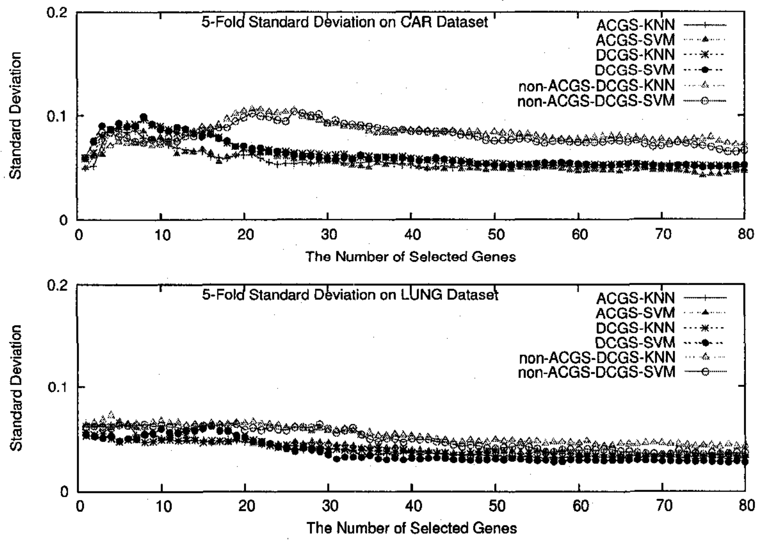


Figure 3.8: Stardand deviation over 300 5-fold classification accuracies of the ACGS-based, DCGS-based and non-ACGS-DCGS based methods, combined with KNN-classifier and SVM-classifier, on the CAR and LUNG datasets, respectively.

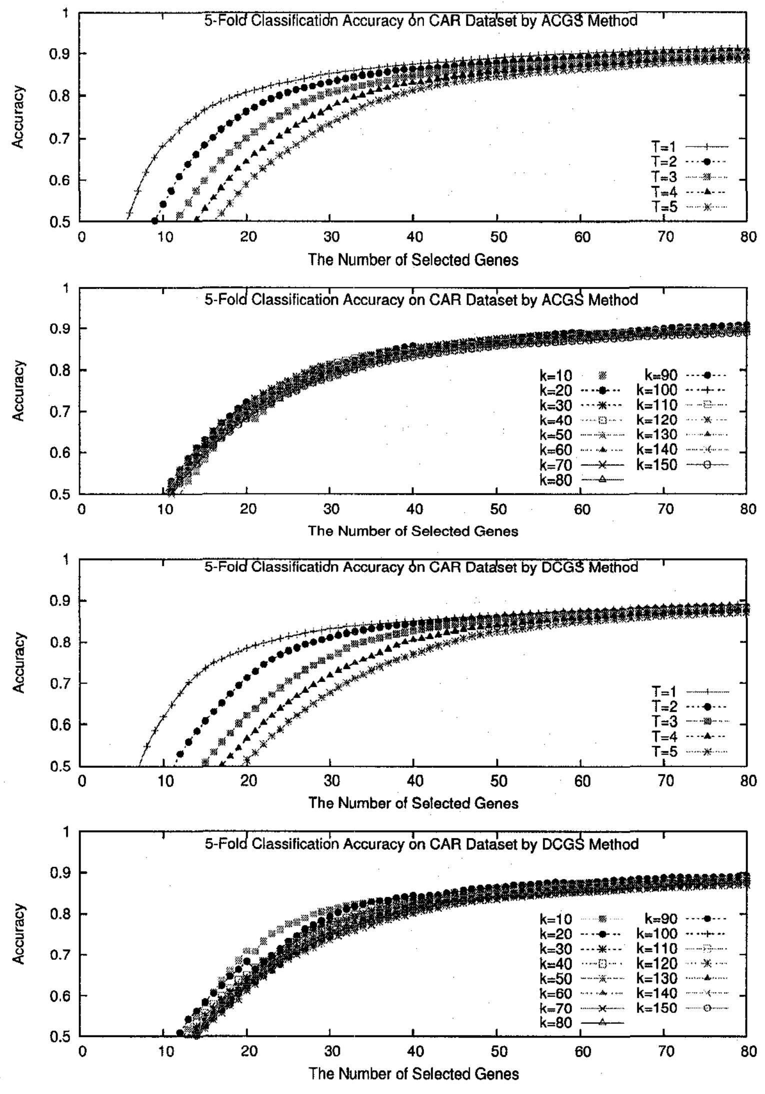


Figure 3.9: Qualities of different value of T and k of the ACGS based and DCGS based methods, tested on the CAR dataset. For a value of T, its quality is the average classification accuracy over 9000 ones; For a value of k, its quality is the average classification accuracy over 3000 ones.

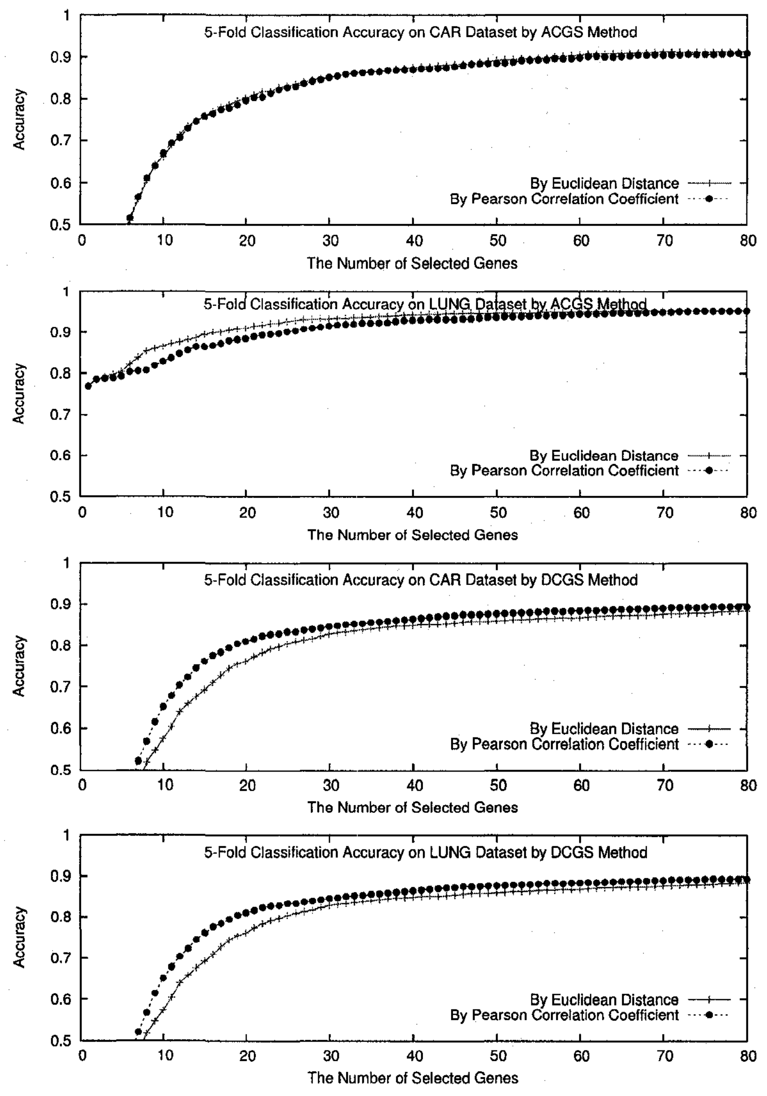


Figure 3.10: Quality of Euclidean distance and Pearson correlation coefficient of the ACGS based method and the DCGS based method, tested on the CAR dataset and the LUNG dataset, where $T = 1$ and $k = 100$.

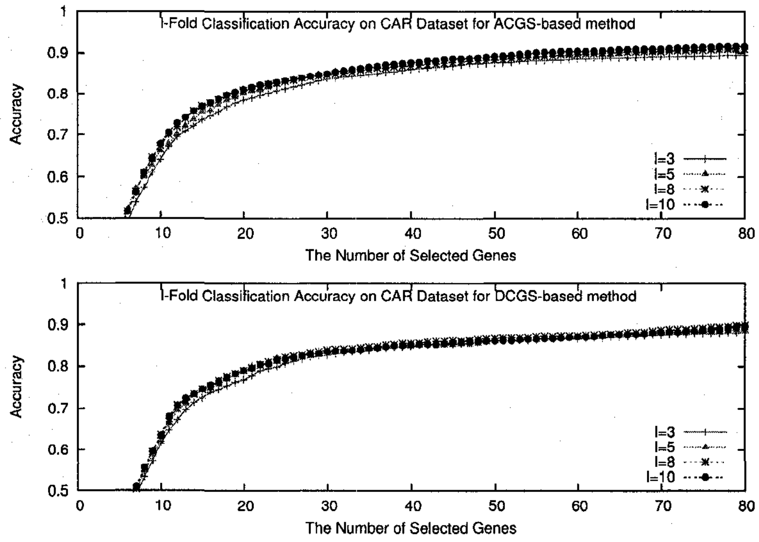


Figure 3.11: Quality of different ℓ values for ℓ-fold classification accuracies of the ACGS and the DCGS gene selection methods, tested on the CAR dataset.

Chapter 4

Smoothing Blemished Gene Expression Microarray Data via Missing Value Imputation

4.1 Introduction

As we mentioned in the previous chapter, microarrays, typically high-density oligonucleotide arrays, such as Affymetrix GeneChip oligonucleotide arrays and Agilent Dual Mode whole genome gene expression arrays, can simultaneously assess expression levels of thousands of genes under a variety of conditions. Such a high-throughput technology provides a unique tool for systems biology, and has important applications in numerous biological and medical studies. One of the most common and important tasks in these applications is to compare the gene expression levels in tissues under different conditions, such as healthy versus diseased, for effective genetic profiling. Missing value is not the only problem preventing us from obtaining the complete accurate data. Even when there are no missing values in the matrix, many entries in the micarray might not be reliable because of noise.

Noise in gene expression microarray data comes from many sources, some of which is caused by experimental setup, such as insufficient resolution, image corruption, or even dust and scratches on the slide [96]. Other noise could be caused by the chip design itself. For example, in general, probes can over- or underestimate gene activity [61]. A probe set in an Affymetrix oligomucleotide array normally consists of multiple pairs of oligonucleotide probes [43]. Using pre-specified mapping criteria, the expression value of the probe set can be obtained from the hybridization levels of these probe pairs. The large number of probe pairs guarantees a substantially low probability of missing all the hybridization levels, thus ensuring reading of the expression value for a probe set. However, although with a certain set of criteria for assessing the overall quality of a chip, probes have not been designed to be specific to gene splice variants and little sensitivity is promised for detecting localized artifacts, such as "harshlight" in microarray image. As pointed out by Suarez-Farinas *et* *al.* [92, 91], about 25% of chips they collected contain artifact blemishes. Unfortunately, so far there are no safeguards to signal potential physical blemishes. Another confounding factor for getting accurate expression data is cross-hybridization. Oligonucleotide probes often relate not only to gene products that exactly match the sequence, but also those with near matches. Furthermore, there are still certain microarray probes targeting almost the same regions of a given gene but giving wildly different intensity signals [61].

On the other hand, all the downstream computational data analyses require the gene expression dataset to be complete and accurate. Therefore, it is desirable to identify the inaccurate data entries (called *stains*), if any, and adjust them. Several ideas have focused on discovering inaccurate data entries. For example, due to ozone degradation, one channel in a two-channel microarray experiment would produce poor quality data. There are two possible solutions: one is to exclude one channel, and the other is to discard only the affected arrays. Lynch *et al.* [60] proposed to combine these two methods with a linear model to detect affected positions. The tests on several datasets showed that it outperformed either individual method. Due to variation in experimental conditions, it is difficult to combine the data from different arrays. Barenco *et al.* [10] proposed a simple recursive algorithm to correct the mismatches in oligonucleotide microarray data to increase the precision of the dataset, by using constant genes to rescale the datasets such that expression data are normalized and consistent. Tran *et al.* [95] presented a new approach to correctly identify accurate signals and used a simple correlation between mean and median to adjust those inaccurate signals. They also demonstrated that their method is better than thresholds or other traditional methods. Blemished data are usually outliers in the dataset, and those caused by different reasons will have different outlier patterns. Suarez-Farinas *et al.* [92, 91] proposed a simple method to find "harshlight" blemishes in chips due to physical or chemical problems. Using statistics on a number of the same arrays under the similar experimental conditions, the authors devised a pattern recognition algorithm to identify and eliminate a variety of defects.

Here we assume complete microarray datasets and present a method to computationally discover the expression outliers as inaccurate data entries, then re-estimate them. We evaluate the quality of the resultant *smoothed* datasets through a downstream application — feature gene selection and the performance of the sample classifier built on the selected feature genes. The rationale supporting such an evaluation is that only an accurate prediction of sample conditions can eventually demonstrate the value of the gene expression microarrays [61]. In further details, first, on the original dataset with labeled samples, we apply a single gene ranking method, F-test [9] (and Scatter [22]), to assign each gene a score. That score represents the discriminatory strength of the gene for distinguishing different sample classes. The data entries which mostly affect the gene scores are regarded as stains. The rationale is that, normally, one gene has similar expression tendencies under the same conditions. When one entry is expressed in an exceptional way, it is likely to be blemished and should be treated as an inaccurate entry. After the stains are detected, a missing value imputation method, KNNimpute [96], can be applied to re-estimate them. We have included three real human cancer gene expression microarray datasets, each obtained using a common platform, in the experiments to demonstrate the success of our method. These three human datasets are the CAR dataset [90], Ovarian dataset [75], and GLIOMAS dataset [66]. We calculated the 5-fold cross validation classification accuracies for the KNN-classifier and the SVM-classifier, which are built on a number of genes selected from the smoothed datasets and the original datasets. The achieved classification accuracies before and after the data smoothing on all these datasets are statistically different, indicating that smoothing is able to at least partially adjust data blemishes.

4.2 Methods

The gene expression data generated from microarray experiments is presented as a matrix $A_{p \times n}$, in which there are p genes, n samples, and a_{ij} denotes the expression level of the i-th gene in the j-th sample.

In the 5-fold cross validation scheme, $\frac{4}{5}$ of the samples are used as the training dataset, in which every sample is labeled by its class membership. To build a sample classifier for testing sample prediction, a gene selection method is used to identify a number of discriminatory genes. In this study, we adopted two existing gene selection methods: F-test [9] and Scatter [22]. Essentially, each gene selection method assigns a score to every gene, and a bigger score indicates a higher class discrimination strength. On the training dataset, for example, F-test assigns a score s_i for gene i. For each entry a_{ij}, F-test method ignores the j-th sample to assign gene i another score s'_i. The value $s_{ij} = |s_i - s'_i|$ measures the abnormality of data entry a_{ij}, and a higher value indicates a more problematic entry. F-test is used to assign an abnormality value to each data entry in the training dataset. Later on, the top few percents of them, ranked by the abnormality values, s_{ij}, and under three separate inaccurate data entry distributions, are identified as *inaccurate*. We note that such a process of inaccurate entry identification relies on the detailed gene selection method, and different methods might identify different sets of inaccurate entries. Nevertheless, F-test and Scatter performed quite consistently in our experiments. Also the dataset should be large enough, that is, a sufficient number of genes and a sufficient number of samples in each class; for otherwise the identification result could be biased.

The discovered inaccurate data entries are erased from the training dataset, i.e., treated as missing. A missing value imputation method is called to impute (or, re-estimate) their values. There are more than a dozen imputation methods proposed in the last decade. In this study, we employed the weighted K-Nearest Neighbor imputation (KNNimpute) method by Troyanskaya *et al* [96] (three other methods, SKNNimpute [48], BPCAimpute [67], and ILLSimpute [18], were also used, whose performance were similar but slightly worse than KNNimpute, data not shown). KNNimpute is shown to be a simple yet competitive missing value imputation method. The imputed (or smoothed) training dataset is complete and is ready for the next step of feature gene selection.

The two gene selection methods, F-test and Scatter, are re-used to select a number of feature genes on the smoothed datasets, for sample classifier construction. We included in this study two classifiers: the K-Nearest Neighbor (KNN) classifier [32] and a linear kernel Support Vector Machine (SVM) classifier [39] (a logic regression (LR) classifier [31] was also used, whose performance was similar, data not shown). In brief, the KNN-classifier predicts the class membership of a testing sample by using the expression values of (only) the selected genes, identifies the K closest samples in the training dataset and then uses the class memberships of these K similar samples through a majority vote. In our experiments, we set the default value of K to be 5, after testing K from 3 to 10. The SVM-classifier, which contains multiple SVMs, finds decision planes to best separate the labeled samples based on the expression values of the selected genes. It uses this set of decision planes to predict the class membership of a testing sample. One may refer to Guyon *et al* [39] for more details of how the decision planes are constructed based on the selected genes.

4.3 Results

We include three real human cancer gene expression microarray datasets in this study. Each dataset is represented as a matrix $A_{p \times n}$, where there are p genes, n samples/chips/arrays, and data entry a_{ij} denotes the expression level of the i-th gene in the j-th sample.

4.3.1 Inaccurate Entry Discovery

We tested three separate assumptions on the distribution of the inaccurate data entries inside the expression matrices. Using the original expression matrix, in which every sample has a known class membership, for each gene i, we calculated its score, denoted as s_i, using a gene score method. In this study, two scoring methods were employed: F-test [9] and Scatter [22]. After that, for each entry a_{ij}, we ignored the j-th sample from the original expression matrix and calculated the score for gene i again, which is denoted as s'_i. Let $s_{ij} = |s_i - s'_i|$. Note that, if the training dataset is large enough and with high quality, the difference between s_i and s'_i should not be large. Therefore, the larger the s_{ij}, the more abnormal entry a_{ij} can be regarded, and the top a few percents (in this study, 1–30%) of them, depending on the distribution assumption, are treated as inaccurate or blemished.

In the first assumption, for each gene, the top a few percents among its n entries (i.e., one row in the expression matrix) are treated as inaccurate; In the second assumption, for each sample, the top a few percents among its p entries (i.e., one column in the expression matrix) are treated as inaccurate; In the last assumption, the top a few percents among all the $p \times n$ entries (i.e., the whole expression matrix) are treated as inaccurate. For simplicity, we call them assumptions on genes, on samples, and on whole dataset, respectively. Under either assumption, the detected inaccurate data entries were erased from the expression matrix and one missing value imputation method, in this study, KNNimpute, was used to estimate their values. The resultant (called *smoothed*) expression matrix is complete, and is ready for gene selection to build sample classifiers. Note that when combining multiple microarray chips into one single dataset for data analysis, one has to take into consideration that the individual chips were exposed in possibly, maybe only slightly, different experimental conditions. Therefore, even after the proper data normalization to tune the multiple chips into a common setup, there could be cases where one gene behaves more abnormally than the other, or one chip behaves more abnormally than the other. The three assumptions on the distribution of inaccurate entries were proposed to examine all these possibilities.

4.3.2 Sample Classification Accuracies

On each gene expression microarray dataset, for each *inaccurate rate* (i.e., the percentage of inaccurate entries), we need to collect 4 classification accuracies, namely, the classification accuracy on the original dataset, and the classification accuracies on the imputed dataset based on uniform distribution assumptions on genes, on samples, and on the whole dataset, respectively. We use *Original*, *Gene*, *Sample*, and *Whole* to denote these 4 classification accuracies, respectively. Notice that we tested the inaccurate rate from 1% to 30%.

We adopted the 5-fold cross validation scheme in this study. The samples in the given dataset are randomly partitioned into 5 equal subsets, 4 of which form the *training dataset* for inaccurate entry discovery and estimation, and subsequently gene selection to build sample classifiers. The 5th subset forms the *testing dataset* in which the sample class labels are blinded to the sample classifiers for prediction. Every subset is rotated to be the testing dataset, and the percentage of correctly predicted samples (true positives) is the classification accuracy associated with this partition. The process is repeated 20 times in our experiments, and the average classification accuracy is the final (5-fold cross validation, omitted in the sequel) classification accuracy. Note that we have used two gene scoring/selection methods, the KNNimpute method, and two classifiers. We concatenate their names to label the classification accuracies. For instance, "Gene-1%-F-test-KNNimpute-KNN" denotes the classification accuracy that is achieved by applying the F-test scoring method to discover 1% inaccurate entries for every gene, using KNNimpute to re-fill their entries, again applying the F-test scoring method to select feature genes to build a KNN-classifier, and running the KNN-classifier for sample class membership prediction.

Figures 4.1 4.2, 4.3 plot the classification accuracies of the F-test-KNNimpute-KNN method on the three human cancer microarray gene expression datasets: GLIOMAS, Ovarian, and CAR, respectively, each assumed 1–4% inaccurate entries under all three distribution assumptions. The standard deviations of those 20 5-fold cross validation classification accuracies on each dataset were all very small compared to the average classification accuracies, almost always less than 0.05 and decreasing with the number of selected genes. On all these three datasets, the classification accuracies achieved on the smoothed datasets are almost always higher than those achieved on the original datasets, when the number of selected feature genes ranges from 1 to 80. More specifically, under the assumptions on whole dataset and on samples, the achieved classification accuracies intertwine a bit, but they are clearly higher than the classification accuracies achieved on the original datasets, particularly when the number of selected genes is large (for example, larger than 40). Under the other assumption on genes, the achieved classification accuracies on the smoothed datasets are mostly lower than those on the first two smoothed datasets, yet slightly higher than those achieved on the original datasets. It is worth pointing out that, the differences between the achieved classification accuracies on the original dataset and the smoothed one vary from dataset to dataset, which might be due to the quality of the original datasets. Overall, from these results on the three datasets obtained by Affymetrix genechips, we may conclude that the inaccurate entry uniform distributions on the whole dataset and on samples might be better than the third assumption on genes. This conclusion might largely correlate with imperfect experimental conditions for collecting the gene expression data.

4.4 Discussion

4.4.1 The True Inaccurate Rate

From the results presented in the last section, we see that assuming an equal percentage of inaccurate expression entries for the whole dataset or for samples is reasonable and gives good computational results. In this subsection, we demonstrate that using our smoothing method can actually estimate the most likely (or *the true*) inaccurate rate for each dataset. For this purpose, we experimented with the assumption that some percent of inaccurate expression entries are distributed randomly in the whole dataset. On all three human cancer datasets, we tested inaccurate rates 1–30% and plotted the classification accuracies in Figures 4.6, when 40, 60, and 80 feature genes were selected for KNN-classifier construction. Though not obviously seen from the plots, the GLIOMAS, Ovarian, and CAR datasets seem to have 23%, 4%, and 2% inaccurate entries, respectively, where the smoothed datasets reach the highest classification accuracies (when 40 or 60 genes, 60 genes, and 40 genes were selected, respectively). These inaccurate rates match well with the previously agreed understanding that the CAR dataset has the highest quality and the GLIOMAS dataset has the lowest quality.

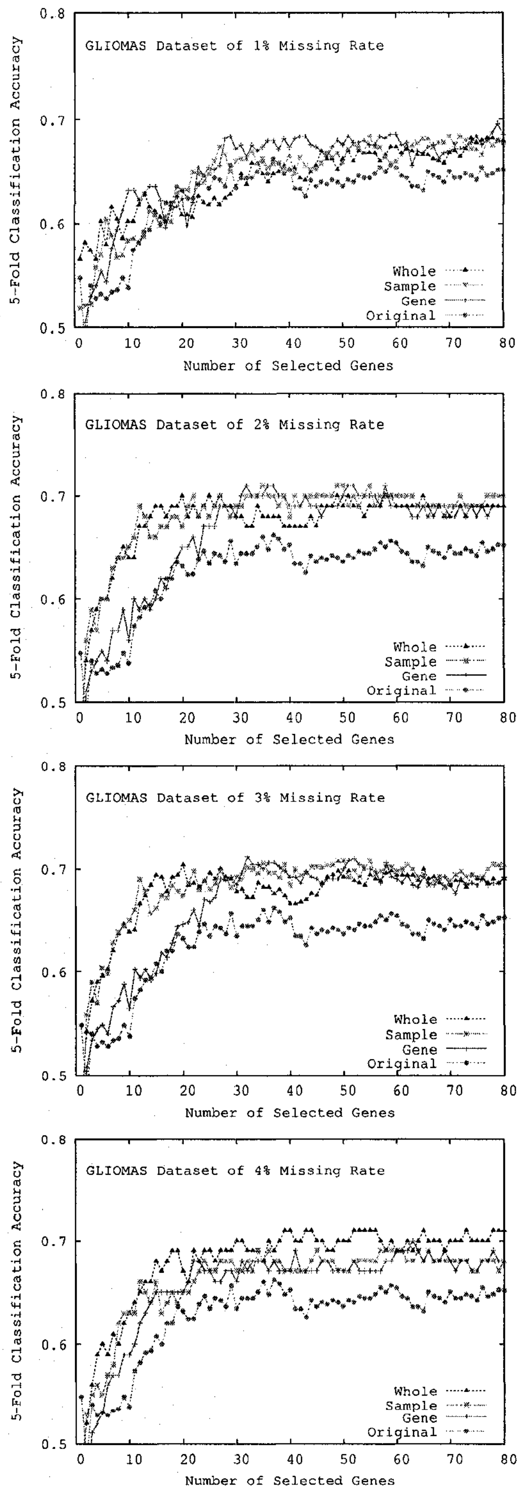


Figure 4.1: 5-fold cross validation classification accuracies of F-test-KNNimpute-KNN on the GLIOMAS dataset assuming 1–4% inaccurate entries under all three distribution assumptions, where 1–80 genes were selected to construct the KNN-classifier.

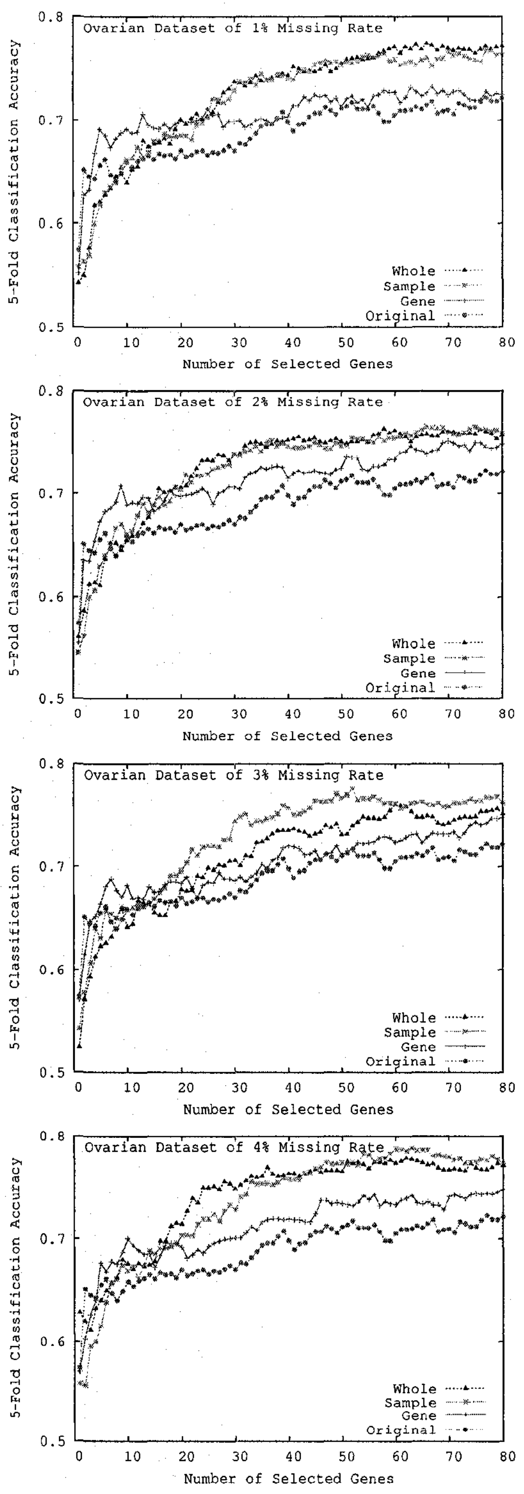


Figure 4.2: 5-fold cross validation classification accuracies of F-test-KNNimpute-KNN on the Ovarian dataset assuming 1–4% inaccurate entries under all three distribution assumptions, where 1–80 genes were selected to construct the KNN-classifier.

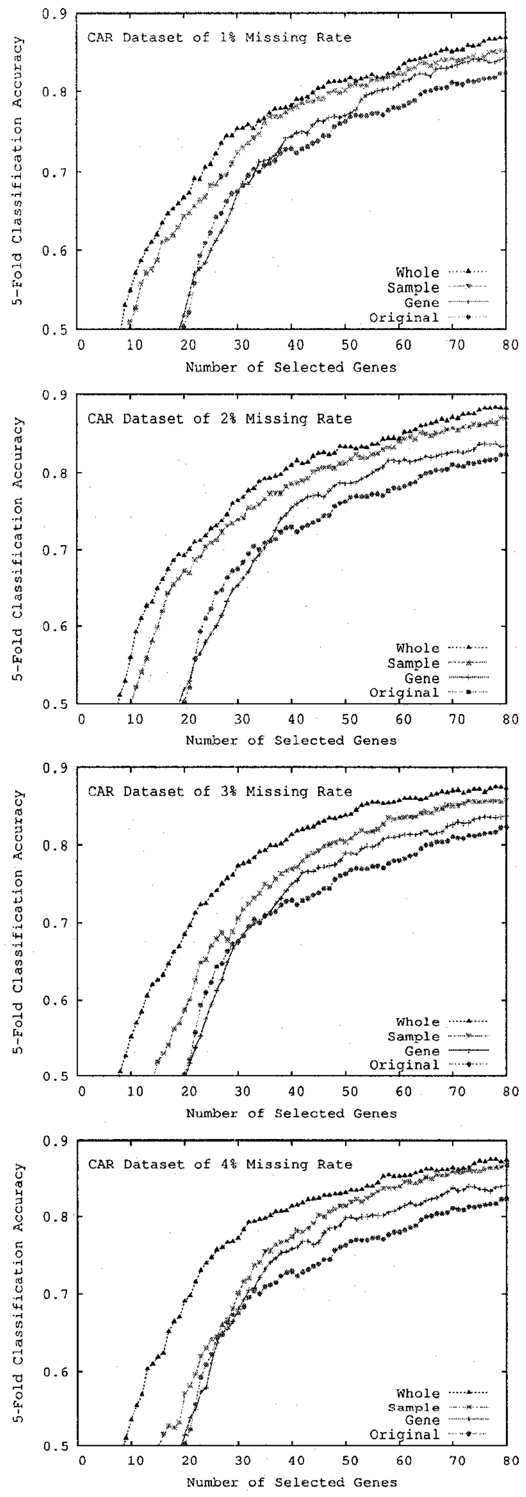


Figure 4.3: 5-fold cross validation classification accuracies of F-test-KNNimpute-KNN on the CAR dataset assuming 1–4% inaccurate entries under all three distribution assumptions, where 1–80 genes were selected to construct the KNN-classifier.

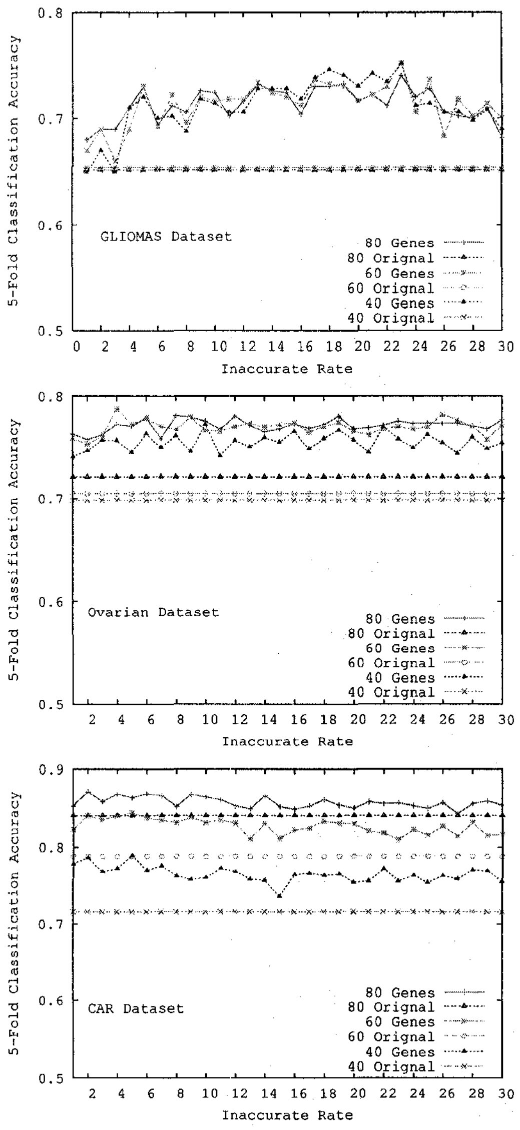


Figure 4.4: 5-fold cross validation classification accuracies of F-test-KNNimpute-KNN on the GLIOMAS, Ovarian and CAR datasets, assuming whole dataset inaccurate rate 1–30%. 40, 60, and 80 genes were selected to construct the KNN-classifier, respectively

Furthermore, on the GLIOMAS dataset, when the inaccurate rate ranges from 1% to 5%, the classification accuracies of F-test-KNNimpute-KNN are plotted in Figure 4.5. For almost every number of selected feature genes (1–80) for building the KNN-classifier, the classification accuracy at 5% is the highest among all. This matches to the first local peak in Figure 4.4.

4.4.2 Difficult Samples Now Correctly Predicted

From Figure 4.4, we see that when assuming 4% whole dataset inaccurate rate in the Ovarian dataset, F-test-KNNimpute-KNN reached the highest classification accuracy (78.75%) when the KNN-classifier was built on 60 selected genes. In the leave-one-out cross validation (LOOCV) to select 60 genes, the achieved classification accuracy by F-test-KNN-classifier is also 78.75%. We

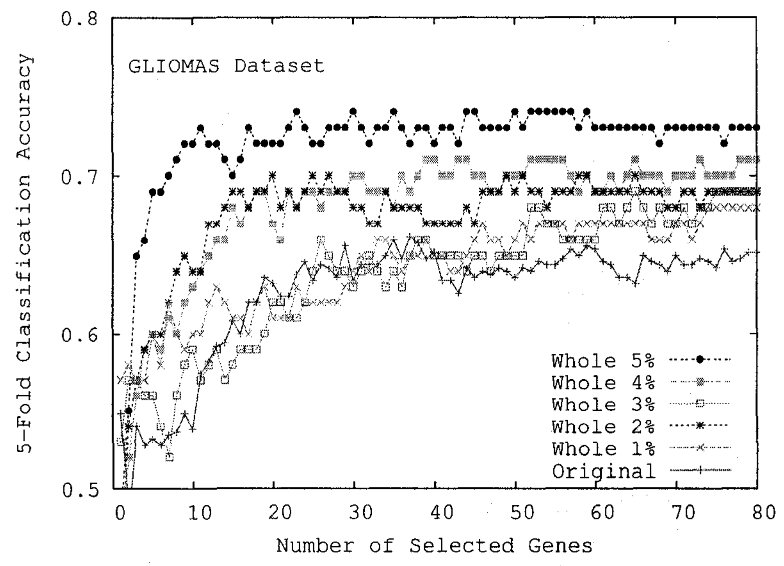


Figure 4.5: 5-fold cross validation classification accuracies of F-test-KNNimpute-KNN on the GLIOMAS dataset assuming 1–5% inaccurate entries in the whole dataset, where 1–80 genes were selected to construct the KNN-classifier.

collected the detailed confusion matrix on the smoothed datasets, and compared it with the LOOCV sample class prediction confusion matrix on the original dataset by F-test-KNN-classifier (78.85% versus 72.12%), in the following Table 4.1. Note that there are 9 mucinous samples and 2 clear cell samples, which were difficult for prediction, now correctly predicted. One also sees that there are 4 serous samples now mis-classified. One possible cause is the gene selection method F-test/Scatter. Note that each gene has its own power to differentiate some specific pairs of classes. In our experiments, only a limited number of feature genes were picked and combined with classifiers to do the classification. Consequently, when some feature genes that have more discriminative power to identify the mucinous and clear cell samples were selected to construct the sample classifier, the genes that have more discriminative power to identify the serous samples might be kicked out. In this case, the prediction on serous samples became worse.

Class	Original				Smoothed @4%			
serous	**51**		2		**47**		6	
endometrioid	1	**5**	4			**5**	5	
mucinous	18		**15**		9		**24**	
clear cell		3	1	**4**		2		**6**

Table 4.1: The detailed LOOCV sample prediction result on the original Ovarian dataset by F-test-KNN-classifier and on the smoothed dataset assuming 4% whole dataset inaccurate rate by F-test-KNNimpute-KNN while selecting 60 feature genes for classifier construction.

4.4.3 NRMSE for Imputed Inaccurate Entries

In the literature, the performance of missing value imputation methods is usually measured by the *normalized root mean squared errors* (NRMSE) defined as

$$\text{NRMSE} = \frac{\sqrt{\text{mean}[(a_{ij}^{\text{imputed}} - a_{ij}^{\text{true}})^2])}}{\text{std}[a_{ij}^{\text{true}}]},$$

where a_{ij}^{true} are those data entries detected to be inaccurate and a_{ij}^{imputed} are the corresponding estimated values by the missing value imputation method. A smaller NRMSE indicates that the imputed values are closer to the original given values (however, notice that in this study these given values are detected to be inaccurate). The KNNimpute method has been shown to achieve average NRMSEs of less than 1.0 [96]. In Figure 4.6, we plotted two series of NRMSEs by F-test-KNNimpute for whole dataset inaccurate rates of 1–30% on the GLIOMAS dataset. In one series (whole separate), the NRMSE at an inaccurate rate the 2% denotes the NRMSE when the inaccurate rate is 2%, the NRMSE at 4% denotes the NRMSE on the next 2% inaccuate data entries, and the NRMSE at 6% denotes the NRMSE on the succeeding 2% inaccuate data entries, and so on. In the other series (whole average), the NRMSE at inaccurate rate $p\%$ denotes the NRMSE when the inaccurate rate is $p\%$, for $p = 2, 4, 6, \ldots$. Clearly, both series of NRMSEs decrease when the inaccurate rate increases, which is another indication that the most inaccurate data entries were picked up first by our method. Furthermore, all the NRMSEs are greater than 1.0, which help confirm that the discovered inaccurate data entries might indeed be noisy. Lastly, the NRMSE peaks around 4%, colliding with the classification accuracy peak at 4% in Figure 4.6, indicating again the positive effects of the smoothed data entries.

Besides plotting the achieved NRMSEs by F-test-KNNimpute on the discovered inaccurate data entries, we have conducted an independent test on the randomly selected data entries, at the same percentages ranging from 1–30%. For each percentage, the random selection process was repeated 10 times, and the NRMSEs at the 2% increment and the average NRMSE by KNNimpute are plotted in Figure 4.6 (labeled with 'Random Separate' and 'Random Average', respectively). Interestingly, all these NRMSEs are very close to 1.0, strongly indicating that the first 12% data entries picked up by our smoothing method are very likely inaccurate.

4.4.4 Experiments on Simulated Datasets

In these experiments, we demonstrate that our smoothing method can indeed discover the inaccurate data entries and smooth them to improve dataset quality. We further demonstrate this through a simulation study, where a good quality gene expression microarray dataset is artificially perturbed with random noise. We examine the performance of a classifier on the original dataset, the perturbed dataset, the smoothed dataset based on the original dataset, and the smoothed dataset based

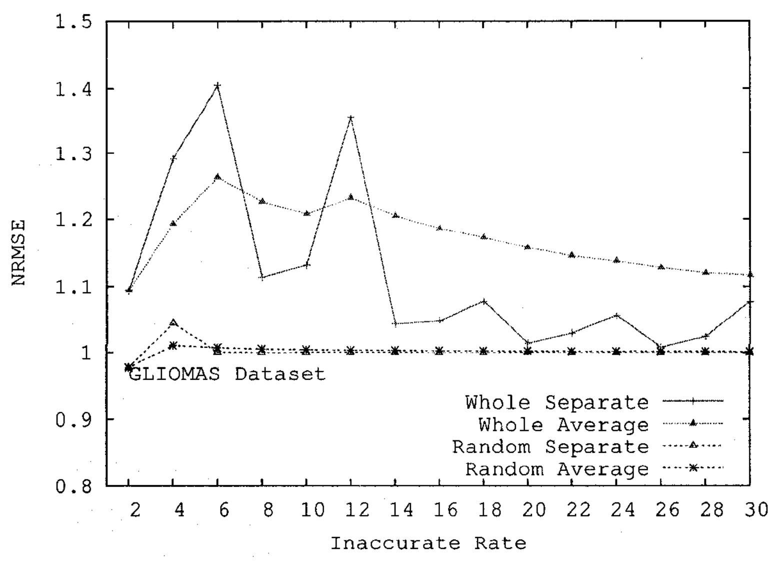


Figure 4.6: NRMSEs of F-test-KNNimpute for whole dataset inaccurate rates the 1–30% on the GLIOMAS dataset, and the NRMSEs of KNNimpute on randomly simulated 1–30% missing data entries.

on the perturbed dataset. We use three datasets: the CAR dataset, the LUNG dataset (U95A oligonucleotide probe arrays), and the SRBCT dataset, in this simulation study. The 5-fold cross validation classification accuracies of an F-test-KNN-classifier on these three (original) datasets are all higher than 90%, and therefore considered as of good quality.

On each of the three datasets, we randomly selected 10% data entries from the whole dataset and perturbed them by adding to them a 0-mean uniform distributed noise with the standard deviation equal to the absolute expression value. The subsequent smoothing method was used to identify the same percent of data entries from the whole dataset, and treated them as inaccurate. Figures 4.7 plots the 5-fold cross validation classification accuracies on the original dataset, the perturbed dataset, the smoothed dataset based on the original dataset, and the smoothed dataset based on the perturbed dataset. To summarize, though varying a little, the performance of F-test-KNNimpute-KNN on the two smoothed datasets is slightly better than on the original dataset, and the performance on the perturbed dataset is significantly worse. Note that these three original datasets are considered as of good quality, and therefore our smoothing method may only contribute a little. However, when the noise is obvious, such as the perturbed datasets, our smoothing method works effectively in discovering the inaccurate data entries and curating them to improve the data quality, which is reflected by the competitive sample classification accuracies, to the ones obtained on the original datasets, achieved by the same classifier.

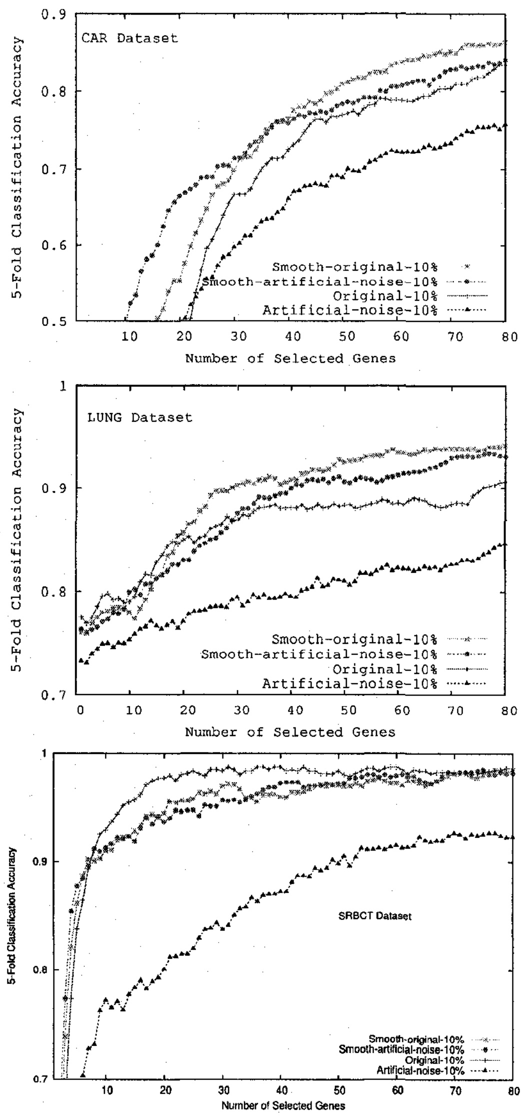


Figure 4.7: 5-fold classification accuracies of F-test-KNNimpute-KNN on the original, the perturbed with 10% whole dataset inaccurate rate, the smoothed based on the original, and the smoothed based on the perturbed, CAR, LUNG, and SRBCT datasets.

4.5 Conclusions

Gene expression microarray datasets are in general noisy. We proposed a novel computational method to detect those inaccurate data entries based on their effects on feature gene selection and a subsequent sample classification. The discovered inaccurate data entries were then re-estimated using existing missing value imputation methods. The extensive experiments showed that the proposed smoothing method reduced the noise level in the original datasets, and that the smoothed datasets had significantly better quality in terms of feature gene selection and sample classification. Another discovery is that, in general we may not be able to assume that every gene has the same probability of being inaccurate, but using our smoothing method to adjust a fixed percent of entries from each sample or from the whole dataset can reduce the noise effect.

Chapter 5

Nucleotide Composition String Selection in HIV-1 Subtyping Using Whole Genomes

5.1 Introduction

[1] The increased volume of available genomic data has made possible phylogenetic analysis for large sets of organisms at the whole genome scale. However, given that most genomes contain millions to billions of nucleotides, traditional molecular phylogenetic analysis approaches based on multiple sequence alignments, such as maximum parsimony and maximum likelihood, become impractical due to their high computational complexity. Moreover, different genes have different evolutionary rates; it has been shown that phylogenetic analyses using different (sets of) genes may give inconsistent results. For instance, for the *human immunodeficiency virus* (HIV-1), the envelope gene is known to evolve much faster than other genes [53]; and for the *Ecdysozoa* clade of animals, the accepted reliability of 18S rRNA as a phylogenetic marker has been questioned [29]. Consequently, it is believed that sophisticated analyses on the whole genome sequences are required to provide a detailed and accurate picture of evolutionary relationships. However, the huge size of these whole genome sequences generally creates computational challenges including memory consumption and CPU usage.

There exist several attempts to address phylogenetic questions from a whole genome perspective, based on efficient information representation of the whole genomes while bypassing the high computational complexity stage of multiple sequence alignments [64, 38, 45, 71, 83, 23, 41, 89, 87, 40, 86, 88]. All of these approaches are intended to extract the hidden evolutionary information from the whole genomes, but from different angles. For example, gene content based methods [83, 82, 41, 42] mainly concentrate on a portion of homologous genes shared by multiple genomes, and then define an evolutionary distance between two genomes based on their gene sharing percentage. Alternatively, the compression based methods [64, 38, 23] generally regard the whole genomes

[1]This work has been published in Bioinformatics [99].

as plain text, and define the similarity between two genomes as the relative compression ratio. The disadvantages of the above two approaches are that the former requires prior knowledge on homologous genes and the latter suffers from aggregate errors arising from compression.

The third class of methods in the whole genome phylogenetic analysis attempt to extend single nucleotide or single amino acid composition to study string composition for whole genomes where a string is a consecutive segment of nucleotides or amino acids [45, 55, 89, 87, 40, 86, 69, 88]. Recent proposals in this category include [45], which analyzed the systematic differences in dinucleotide frequencies within and between species, and obtained a biologically plausible phylogenetic tree for mitochondrial genomes, [55, 40, 69], in which analyzed asymmetries in length-k word distribution, then extracted phylogenetic properties from genome-wide statistical observables for prokaryotes, and [89, 87, 86, 88], which used singular value decomposition (SVD) to analyze short peptide frequencies (of length 3 to 5), then built species phylogenies.

In the reported experimental results, all of the above mentioned methods in the third category managed only strings of length 7 or less for amino acid sequences and of length 12 or less for nucleotide sequences into computation, because of memory demands. Theoretically, one may increase the maximum string length by having finer composition for the whole genomes in order to obtain more accurate pair-wise evolutionary distances. However, increasing string length requires too much memory to be practical, as well as increased CPU usage. For example, computing a length-7 peptide composition for a whole genome (which is regarded as the union of its encoding proteins) already requires gigabytes of storage, regardless of the size of the genome. Consequently, in practice, the maximum string lengths have been set to relatively small values such as 5 and 6.

Nevertheless, it has also been observed that not every composition string contributes equally to the evolutionary distance calculation. In fact, some strings appear to have more discriminatory power than others. [89, 87, 86, 88] proposed to employ SVD on the peptide-to-genome frequency matrix, to extract a reduced number of string linear combinations, and then use their pseudo frequencies to represent the genomes. However, such a decomposition does not address the memory issue, *i.e.*, the peptide-to-genome frequency matrix must still be computed, and that can only be done when the maximum string length is a small value. In addition, the string linear combinations created by SVD are difficult to explain biologically.

Based on the above two major observations, we propose a string scoring method to extract explicit composition strings that heuristically identify the richest evolutionary information, and then to use only these selected strings in the evolutionary distance calculations. These selected composition strings can be regarded as the most important features with respect to the whole genomic sequences. In our method, the number of selected strings is a parameter that can be tuned depending on the available computing resources. In particular, the memory requirement in the selection process is proportional to the number of selected strings, and selecting thousands of strings can be processed on a normal desktop, while examining candidate strings of an arbitrarily large length.

We applied our method on a dataset of 867 pure subtype HIV-1 strains and 331 various recombinants, to predict their subtypes or recombinant forms. Among the 867 pure subtype strains, 42 were used as references. By setting the number of selected strings at 500 and the maximum string length at 21, we achieved 100% leave-one-out subtyping accuracy on the reference dataset of 42 pure subtype strains. Using these 500 strings, we also achieved 100% subtyping accuracy on the independent testing dataset of the other 825 pure subtype strains. These 867 pure subtype strains were also used in blind comparison to three most recently proposed HIV-1 subtyping programs [65, 25, 70], which achieved 96.4%, 99.2%, and 99.5% accuracy, respectively. More detailed analysis revealed these 500 top scoring strings to be signature strings associated with certain subtypes. Subsequently, we present a method to remove 2–50% of consecutive nucleotides from each of the 331 recombinant strains and then to predict the subtype information for the remaining sequence. The non-trivial percentage (for example, 3%) of predicted subtypes match well with the known recombinant forms, with some exceptions strongly suggesting the need for further human re-curation. All these results demonstrate that our proposal is promising in terms of both the biological significance of the selected nucleotide composition strings and the quality of the recovered phylogenetic relationships.

The rest of the chapter is organized as follows: In the next section, we briefly introduce the *Complete Composition Vector* (CCV) representation for a whole genome. We will then present a selection scheme to extract the most informative nucleotide composition strings. Using these selected strings, we can obtain a much lower dimensional composition vector for each genome. We then define the evolutionary distance between two genomes based on their composition vectors. In Sections 5.3 and 5.4, we report and discuss the computational results on the HIV-1 subtyping, respectively. Section 5.5 presents the recombinant form prediction and the preliminary experimental results. We conclude the chapter in Section 5.6.

5.2 Methods

5.2.1 Complete Composition Vector

We use whole genomic sequences to introduce the concept of CCV. For a genome represented as the union of its encoding protein sequences, its CCV can be analogously defined. First, a length-k string is a sequence of k consecutive nucleotides. Given a whole genomic sequence G of length L, the number of appearances of a length-k string $\alpha = a_1 a_2 \ldots a_k$ in G is $f(\alpha)$, where every a_i is a nucleotide. Since there are $L - k + 1$ (overlapping) length-k strings in G in total, the *probability* of appearance of string α in sequence G is $p(\alpha) = f(\alpha)/(L - k + 1)$. Similarly, we can define the probability of appearance $p(\alpha)$ for string α in a whole genome containing multiple chromosomes, where the dividend becomes the number of appearances across all the chromosomes and the divisor becomes the total number of (overlapping) length-k strings in all the chromosomes.

Based on all the string appearance probabilities we can define the composition value $\pi(\alpha)$ for string α. In this chapter, we adopt a second order Markov model similar to [40]. In such a model, we first calculate the expected appearance probability of string $\alpha = a_1a_2 \dots a_k$ as $q(a_1a_2 \dots a_k) = (p(a_1a_2 \dots a_{k-1}) \times p(a_2a_3 \dots a_k))/p(a_2a_3 \dots a_{k-1})$, and then define the composition value $\pi(\alpha) = (p(\alpha) - q(\alpha))/q(\alpha)$. All the composition values are stored in a sequential order to form a vector $V_k(G) = \langle \pi_1, \pi_2, \dots, \pi_m \rangle$ that represents the whole genome G, where k is the string length and m denotes the total number of strings under consideration. In [40] and [69], the (amino acid) string length k was fixed at a very small single value (less than or equal to 6). In one of our previous research [100, 99], we conducted a systematic study and concluded that using strings with multiple lengths, in a range $[1, K]$ for some K, is more effective. Particularly, the phylogenetic analysis and the resultant phylogeny in [100] showed improvements over using only one fixed length. In this chapter, we continue to use strings of length in the range $[1, K]$, and the vector definition by all these composition values of strings, *i.e.*, the concatenation of $V_1, V_2, \dots, V_K$, is referred to as the *Complete Composition Vector (CCV)* of the whole genome. Certainly, a larger value of K gives a vector containing finer evolutionary information.

A CCV is thus an m-dimensional vector (for instance, m could be as large as $4 + 4^2 + 4^3 + \dots + 4^{15} = 1{,}431{,}721{,}300$, when $K = 15$). Note that m could be a very large number, and it implies one major disadvantage of CCV for acquiring too much memory to be computationally efficient. In the preliminary experiments, we set the target to examine strings of length up to 100, and therefore the memory issue needs to be addressed. Firstly, observe that there are strings, especially when they are long, which do not occur at all in any whole genome in the dataset. We thus do not compute their composition values. Subsequently, the CCV for a whole genome has a much lower dimension, in which every entry is associated with a string that occurs in at least one genome. Secondly, notice that not all strings contribute to the phylogenetic analysis equally. Therefore, we propose to extract only a small number of strings, which contain the richest evolutionary information, and only use them in the phylogenetic analysis. The proposed string extraction scheme is based on the measurement of *relative entropy*, which has been constantly employed in the general feature selection in the statistical learning literature. The most important parameter in this framework is the number of extracted strings, which is likely dataset dependent and, on the HIV-1 subtyping, is set at 500 through extensive preliminary/training experiments. Such a setting is to maintain the overall quality of the recovered HIV-1 phylogenetic relationships. Under this string extraction scheme, we were able to examine much longer strings (in our experiments, up to 100) without causing any memory problems, and as a result we discovered that those composition strings with the richest evolutionary information in HIV-1 subtyping have length mostly in the range $[5, 9]$. Such a discovery partially confirms the idea that we can skip long strings in the whole genome phylogenetic analysis. The reported HIV-1 subtyping results are on strings of length 1 to 21. It also confirms that using a single length is not sufficient [100], and thus the CCV representation is in general more effective than the representation proposed in [40] and [69].

5.2.2 String Selection and Phylogenetic Relationships

In this section, we first introduce a scoring scheme to estimate how important a nucleotide composition string is, and then, by selecting the top ranked 500 strings, we obtain a 500-dimensional composition vector for each whole genome. Two other scoring schemes that have also been tested and the empirical determination of the string number 500 are included in Discussion. On the HIV-1 dataset of 42 reference strains, we note that 500 is much smaller than the total number of examined strings (of length 1 to 21), which is $2,260,957$, and these 500-dimensional vectors can be computed without causing any memory problem. Note also that, disregarding the memory issue, the string selection is done in almost the same amount of time for computing the CCV representation, except a negligible amount of time for computing relative entropies. These vectors are then used to define a pair-wise evolutionary distance between every pair of whole genomes, and then the achieved distance matrix is used to construct a Neighbor-Joining [72] phylogeny, and in subtyping. The quality of the recovered phylogenetic relationships, represented in subtyping accuracy, demonstrates the success of our method and the quality of the selected composition strings.

5.2.3 String Scoring Scheme: Relative Entropy

Basically, the scoring scheme is set up to evaluate the information content associated with the composition strings, and to assign a higher score to a string if its information content is richer. Note that each string is evaluated independently. To begin, we concatenate all the given whole genomes in the dataset and regard the result as a *super-genome*. We then compute the composition value $\pi(\alpha, i)$ for string α in genome i, for each $i = 1, 2, \ldots, n$ (here n is the number of whole genomes in the dataset), and the composition value $\Pi(\alpha)$ for string α in the super-genome.

The absolute composition values $|\pi(\alpha, i)|$ for string α in all the given genomes may be regarded as an *unnormalized* probability distribution of string α, where the index i is regarded as a variable. We use *relative entropy* (or Kullback-Leibler distance) to assign a score to string α to measure the distance between this distribution and the *unnormalized* background probability represented as $\Pi(\alpha)$. Namely,

$$s(\alpha) = \sum_{i=1}^{n} |\pi(\alpha, i)| \ln \left| \frac{\pi(\alpha, i)}{\Pi(\alpha)} \right|,$$

where ln( ) is the natural logarithm. Note that relative entropy is used to estimate the distance between two probability distributions. Therefore, $s(\,)$ defined in the above is close to 0 if the actual distribution is close to the background one. In other words, the larger the absolute relative entropy, the more informative string α is.

5.2.4 Selected String Composition Vector

We maintain a buffer of size 500 to store the nucleotide composition strings that have been examined, and have the highest scores using the above relative entropy based scoring scheme. We examine the strings in increasing length and, for each length, in lexicographical order. For each string under consideration, if there is a room in the buffer (*i.e.*, among the first 500 strings), it is appended; otherwise, its score is compared with the minimum score of the strings stored in the buffer, and if larger, it replaces the string with the minimum score. By only saving these 500 highest scored strings, the potential memory issue is resolved and the maximum string length to be examined can be set to an arbitrarily large value. For example, we have examined strings of length 100 in our preliminary experiments on a normal desktop with 1GB of memory. After all strings have been examined, the composition values of the 500 top scored strings stored in the buffer are used to assemble the 500-dimensional composition vectors to represent the whole genomes. Let $V(i) = \langle \pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_m} \rangle$ be the vector representing genome i, for $i = 1, 2, \ldots, n$, where $m = 500$ and π_{i_j} denotes the composition value of the j-th highest scored string in genome i, for $j = 1, 2, \ldots, m$.

5.2.5 Whole Genome Phylogenetic Relationships

The pairwise euclidean distances are calculated first. The distance matrix $D_{n \times n}$ is used as input to the Neighbor-Joining algorithm to display the phylogenetic relationships among the whole genomes. These distances are also used for HIV-1 subtype prediction for each testing strain. Essentially, the distances between the testing strain and the carefully chosen 42 HIV-1 reference strains are calculated using the above steps of operations, and based on them the subtype or the recombinant form of the testing strain is inferred.

5.3 Computational Results

5.3.1 Overview

To evaluate the effectiveness of our string selection method, we have tested it on a dataset of HIV-1 pure subtype and recombinant strains to predict their subtypes or recombinant forms.

HIV is among the most genetically variable organisms known. HIV-1 is classified into three major phylogenetic groups, M (major), N (new) and O (others). Group M, which is responsible for the HIV pandemic, is further divided into nine subtypes, some of which have been even further subdivided into sub-subtypes. Besides GenBank, there are several other viral databases holding HIV virus sequences, such as the one provided by the Los Alamos National Laboratory (http://hiv-web.lanl.gov/content/index). In 2005, a set of 42 reference whole genomic sequences was published [53], which included 35 sequences in group M, 3 in group N, and 4 in group O. In addition, we have downloaded a total of 825 other pure subtype and 331 recombinant HIV-1 whole genomes for independent testing (several hundred other incomplete strains were excluded from our experiments).

Accurate determination of the genetic subtype for an HIV-1 strain is of crucial importance for epidemiological monitoring as well as for the design of molecular detection systems and potential vaccines [70]. This work addresses mainly the pure subtype HIV-1 subtyping. A discussion on using

length:	4	5	6	7	8	9	10
top 500:	-	2.2	10.0	58.2	22.8	6.6	0.2
top 5,000:	0.138	1.308	7.108	22.85	37.122	24.006	7.462

Table 5.1: Percentages of different length strings in the top ranked strings.

this subtyping system to determine the recombinant form for a recombinant strain is included in Section 5.4. Current subtyping and recombinant form determination methods mostly rely on multiple sequence alignments [65, 25, 62], except the one by [70] based on BLAST search [7]. To the best of our knowledge, multiple sequence alignments have limited quality and are constrained by the size of the dataset (for example, the EMBL-EBI ClustalW server at "http://www.ebi.ac.uk/clustalw/" accepts datasets containing no more than 500 sequences). On the set of 42 HIV-1 reference strains, we selected the 500 top scored strings by relative entropy, and the leave-one-out subtyping accuracy using only the 500 selected strings was 100%. Using these 500 strings, independent subtyping of the 825 testing strains achieved also 100% accuracy. The combined dataset of 867 pure subtype strains were also sent to three other HIV-1 subtyping programs [70, 65, 25] for comparison purpose. Overall, our method successfully avoids the computationally intensive alignment phase, and achieves high subtyping accuracy. Another advantage of our method is that it does not require any pre-knowledge about the genomic sequences (such as that one portion of the genome is more important than the other portion during the subtyping), while those important regions will be automatically detected according to their coverage by the selected strings, which also allow biological explanation.

5.3.2 Results

We applied our string selection method on the set of 42 HIV-1 reference sequences, which can be viewed as the training stage, to select the 500 most informative strings for subtyping purpose. The discerning power of these strings is evaluated through the leave-one-out cross validation on the 42 reference sequences and an independent testing on the dataset containing 825 pure subtype HIV-1 viral sequences. The 42 HIV-1 reference sequences consist of 6 subtype A (4 A1 and 2 A2), 4 subtype B, 4 subtype C, 3 subtype D, 8 subtype F (4 F1 and 4 F2), 3 subtype G, 3 subtype H, 2 subtype J, 2 subtype K, 3 type N, and 4 type O. The average length of these strains is being 9,005bp, with the maximum length being 9,829bp and the minimum length 8,349bp. These HIV-1 reference sequences were carefully selected by considering several criteria [53].

We set a maximum string length K in our method, and the method examined the strings in increasing length and, for each length, in lexicographical order. When $K = 21$, the 500 top ranked (out of $2,260,957$) strings have their length distributed in between 5 and 10. The second row of Table 5.1 shows the percentages of different length strings among these 500 top ranked strings, where one can see that length-7 and length-8 strings are dominant (81.0%). We have also collected the relative entropies of the top $5,000$ strings (whose percentages are in the third row of Table 5.1)

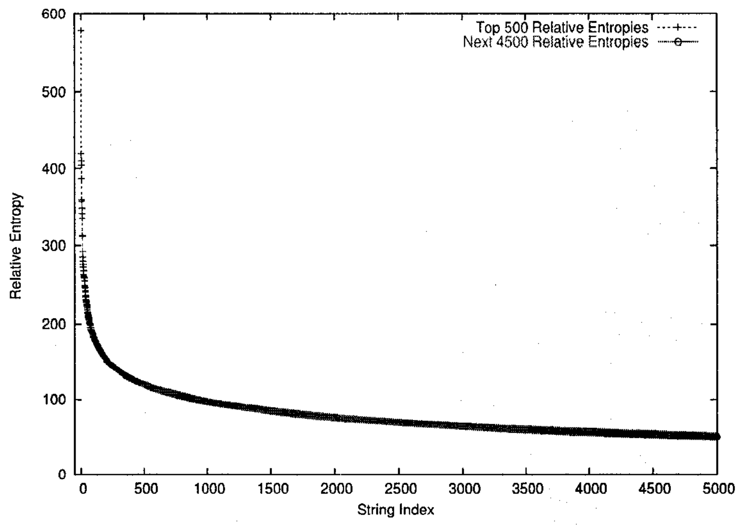


Figure 5.1: The relative entropy scores of the 5,000 top ranked strings, in decreasing order, in which the 500 top ones are colored blue.

and plotted them in Figure 5.1 in non-increasing order. The top 500 strings are colored blue (the other 4,500 in red) in Figure 5.1. From the plot, it is clear that the strings that did not make it into the list of 500 have relatively small relative entropies, and thus can largely be ignored. By representing each strain as a 500-dimensional vector, the subsequently computed evolutionary distance matrix for this set of 42 HIV-1 reference strains was used as input to the Neighbor-Joining method to generate a phylogenetic tree (Figure 5.2), using one CIV strain AF447763 as an outgroup. In this tree, all subtypes are clearly grouped together as distinct branches, and the closeness relationships among the subtypes are also well demonstrated, for example, subtypes B and D are closer to each other than to the others and subtype F (A) indeed contains two distinguishable sub-subtypes F1 and F2 (A1 and A2, respectively).

On these 42 HIV-1 reference sequences, we adopted the leave-one-out cross validation (LOOCV) scheme to predict the subtype information for each sequence whose subtype was blinded. In more details, the testing sequence was removed from the dataset, and the above string selection procedure was applied to the rest of the 41 sequences to identify the 500 top ranked strings by their relative entropies. Note that these 500 strings could slightly differ from the 500 top ranked strings by using all 42 sequences. Next, using the selected 500 strings, the distances between the testing sequence and all the 41 reference sequences were calculated, and the subtype of the closest reference sequence was taken as the predicted subtype of the testing sequence. We repeated this training-testing on each of the 42 sequences and obtained a 100% subtyping accuracy.

Using these 42 reference sequences as a training dataset to select 500 strings, we independently predicted the subtype for each of the 825 pure subtype HIV-1 sequences. Among the 825 sequences, there are 55 A1, 9 A (not known to be A1 or A2), 264 B, 415 C, 51 D, 2 F1, 10 G, 2 N, and 17 O. For each of the testing sequences, whose subtype was blinded, the distances between it and all

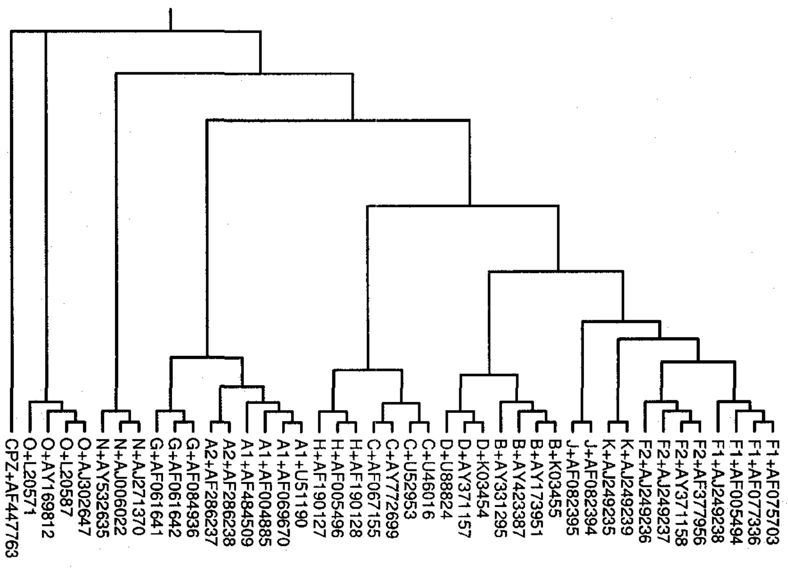


Figure 5.2: The Neighbor-Joining phylogenetic tree on the 42 reference sequences using the 500 top ranked strings, one CIV strain `AF447763` is used as an outgroup.

the 42 reference sequences were calculated using (*only*) the 500 selected strings. The subtype of the closest reference sequence was then taken as the predicted subtype of the testing sequence. We also achieved 100% subtyping accuracy (for each of the 9 A sequences, both A1 and A2 were counted as correct prediction) on this independent testing dataset. Moreover, for each testing sequence, we have observed that the second closest reference sequence from the 42 reference sequences has the same subtype as the closest one. This certainly indicates prediction of high confidence. For each testing sequence, we have also calculated its average distances to all the 13 subtypes. The closest subtype by average distance is exactly the same as the subtype of the closest sequence. Let d_1 and d_2 denote the shortest and the second shortest average distances, respectively, and d_{13} denote the longest average distance. Numerically, we assigned $(d_2 - d_1)/(d_{13} - d_1)$ (*Dixon metric*) as the quantified confidence associated with the subtype prediction. For all the 825 testing sequences, their subtype prediction confidences are plotted in Figure 5.3, in non-increasing order, where only 5 of them are less than 0.1 (0.099233, 0.098523, 0.094155, 0.073713, and 0.052269), which is the normal lower-bound for high confidence [90]. A closer look at these five sequences tells that 1) four of them are of subtype D, and their average distances to subtypes D and B are very close to each other; 2) the other one (`AY173955`) is of subtype B, whose average distance to subtype D is very close to its average distance to subtype B.

To compare with existing HIV-1 subtyping programs [70, 65, 25], we have uploaded all the

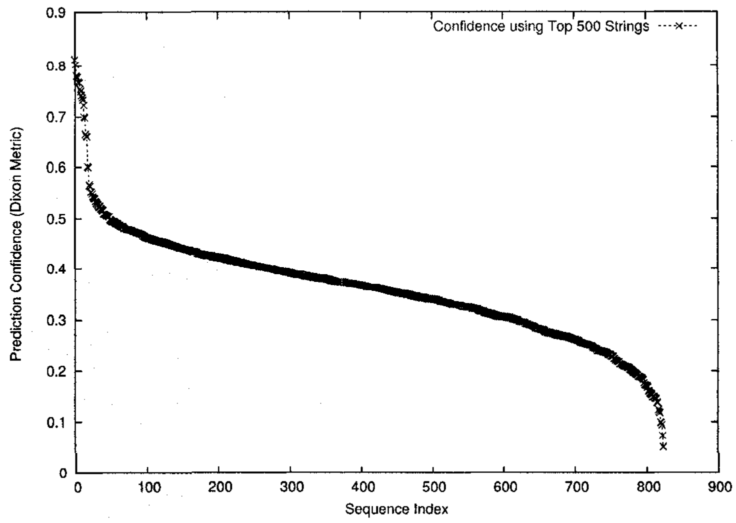


Figure 5.3: Subtype prediction confidence values (Dixon metric) in non-increasing order, using the top 500 strings. Only five out of the 825 predictions are considered not-so-confident under Dixon metric.

867 pure subtype sequences to them to predict their subtypes. The genotyping tool by [70] slides a window along the query sequence and BLASTs each window segment against reference sequences. Similarity scores to reference sequences are returned for each BLAST, and we applied the naive pure subtype assignment using the subtype of the reference sequence with the highest average similarity score. Its overall prediction accuracy was 99.5% (4 were predicted incorrectly, precision 99.5%), but when we forced it to predict pure subtypes only, its accuracy reached 100%. The subtyping system, BioAfrica, by [25] (http://www.bioafrica.net/subtypetool/html/) consists of a multiple sequence alignment by ClustalW, maximum likelihood phylogenetic analysis by PAUP followed by bootscanning, and subtype determination by Treepuzzle. Its prediction accuracy was 99.2% (7 were unassigned, no false positive). The STAR subtyping system by [65] (http://www.vgb.ucl.ac.uk/starn.shtml) evaluates the query sequence against subtype profiles and returns discrimination scores, which are then transformed into a Z-score distribution for determination of HIV-1 subtypes. Its prediction accuracy was 96.4% (31 were unassigned, no false positive). There is a recent independent assessment [36] of three automated genotyping tools including the above BioAfrica and STAR, which (was brought to our attention during the revision) shows many inconsistent genotyping results and suggests that those unassigned strains/sequences show some evidence of recombination. The 867 strains have been annotated as pure subtypes, and our method did perform well, though it might seem aggressive on the not-so-confident prediction.

5.4 Discussion

We mentioned that the maximum string length K to be examined did not affect the subtyping performance in our experiments, as long as it is larger than a certain value C. On the set of 42 HIV-1

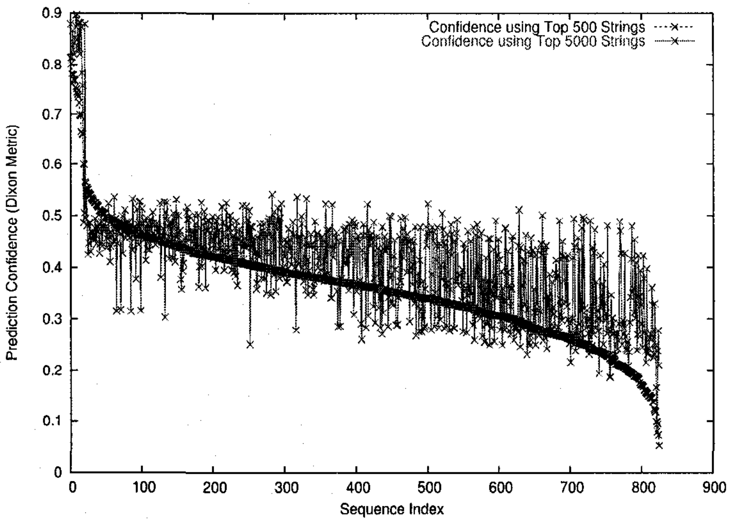


Figure 5.4: The prediction confidence values using the top $5,000$ strings plotted on top of the order by using the top 500 strings. Only one prediction using the top $5,000$ strings remains not-so-confident.

pure subtype strains, C is 10 when we set the number of strings to be selected to $5,000$ or less. Nevertheless, C is clearly associated with the number of selected strings, and a larger number of selected strings would imply a larger value of C. We also believe that C is dataset dependent. That is, for other whole genomic sequences, C could have different values, even when the number of selected strings is the same. We will be investigating this idea on the Avian Influenza Virus (AIV) and Foot and Mouth Disease Virus (FMDV).

To address whether 500 top ranked strings by relative entropy are appropriate for HIV-1 subtyping, we examined a range of selected strings from 50 to $5,000$, at the increment of 50, and checked the corresponding subtyping accuracy. We have observed the first 100% subtyping accuracy at 450, and for every other number tested afterwards, the subtyping accuracy remained at 100% (that is, never decreased). We therefore decided to set 500 as the default. Nevertheless, we have found that, by using $5,000$ strings, most of the subtyping confidences increased. In particular, four not-so-confident predictions using only 500 strings became confident when using $5,000$ strings, and only one (DQ054367) remained not-so-confident (the Dixon metric decreased, strangely, from 0.098523 to 0.080143). On top of the non-increasing order of prediction confidences using 500 strings (blue), the prediction confidences using $5,000$ strings (red) are plotted and shown in Figure 5.4, where one can see that the relative confidences remain largely unchanged and, for most of the testing sequences, the associated prediction confidences using $5,000$ strings increase.

Next, we examined how well the selected 500 strings cover the positions in the HIV-1 whole genomes. For each of these 500 selected strings, if it occurs in one of the 867 pure subtype sequences, then the positions where the string occurs receive a probability of $k/(L-k+1)$ each, where k is the string length and L is the sequence length. The probability that one position receives is regarded as the *coverage probability* of the position, which indicates the relative significance of the position for subtyping purposes. For each position in the multiple alignment of the 42 reference sequences by ClustalW (which was constructed through the EMBL-EBI server at "http://www.ebi.ac.uk/clustalw/", in 113 minutes), we computed its coverage probability and plotted them in Figure 5.5. One can see from this plot that the most frequently covered positions by these 500 selected strings match very well with the most important positions in the ClustalW's multiple alignment (shown as red +, others as blue circles). This is another indication that our method is able to capture the critical sequential evolutionary information indicated by multiple sequence alignments.

In addition to the above string composition values computed within the second order Markov model, we have also examined the first order Markov model in which the composition value directly uses the string occurrence frequency. Also, in addition to the relative entropy scoring schemes, we have tested two other scoring functions: the standard deviation of the composition values and the mean-weighted variant. It turned out that the relative entropy scoring scheme performed significantly better than the other two (data not shown); and the first order Markov model appeared much inferior to the second order Markov model (data not shown). Note that our string selection was done by examining all the strings within the length range, but assuming no correlations amongst them. One immediate future task is to consider possible correlations, and if identified, to borrow ideas from general feature selection methods and classification methods to exploit the feature correlations. Finally, but not of least interest, we will be working with other groups to further investigate the biological content of these 500 selected strings for post-subtyping studies. Interestingly, a BLAST search seemingly shows that the top ranked string GAAAAAGAG, by relative entropy, seems a signature of subtypes B, F, and K (GAAAAAGAG appears in 41.8%, 80%, 100% of subtypes B, F, K strains in our dataset, respectively).

5.5 Recombinant Form Prediction

There are many proposed methods and programs which address RNA recombination detection, especially in RNA viruses such as HIV-1 and *hepatitis C virus* (HCV) [63, 70, 62]. Most of these methods and programs start with multiple sequence alignments [63, 62]. For an HIV-1 viral strain known to be a pure subtype, its subtyping is considered relatively easy. This has been demonstrated by our method, as well as the three subtyping programs we have tested. However, HIV-1 is notorious for its various forms of recombinations, which constantly challenge the drug development [70]. Thus, upon the arrival of each new strain, the subtyping task is to determine whether it is a pure subtype strain, and, if it is not, to determine its recombinant form.

We randomly selected 16 out of the 825 pure subtype sequences, and for each of them, we partitioned it into 50 equal parts, each containing around 180 nucleotides. At each testing, a number of consecutive ℓ parts, where $1 \leq \ell \leq 25$, were removed from the whole strain and the remainder

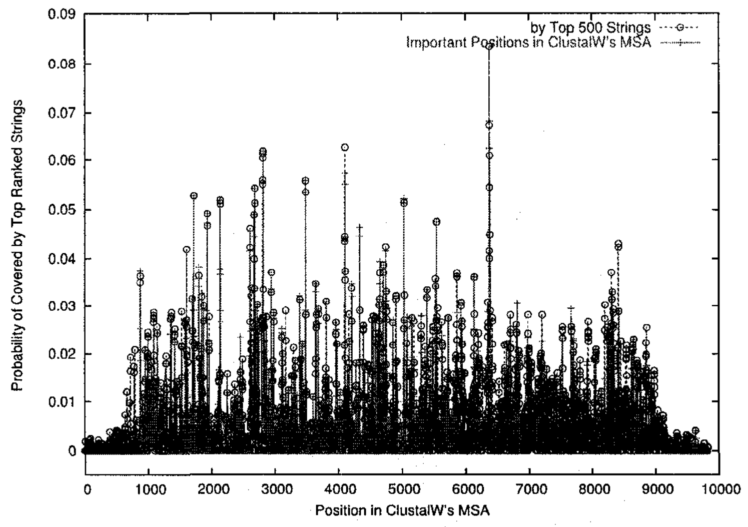


Figure 5.5: ClustalW's MSA position coverage by the 500 top ranked strings, for the 42 reference sequences.

were concatenated into a new sequence. Using the 5,000 top ranked strings selected using the 42 HIV-1 pure subtype reference sequences, the new sequence was again represented as a 5,000-dimensional composition vector. The distances between this vector and the 42 reference sequences were then calculated, and the subtypes of the two closest reference sequences were reported. For each strain, a total of 950 testings were executed and 1,900 predicted subtypes were reported. For each of the distinct 13 subtypes, its occurrence frequency in these 1,900 predicted subtypes was calculated, and every strain was represented as a 13-dimensional vector. These frequencies are color-coded and plotted in Figure 5.6 (the left 16 columns), where one can easily see that those non-B strains are confidently evaluated to be pure subtype strains, though a few of them (two A1, two B, and two D) have been mixed with some small percentage of information from other subtypes.

We then used the above approach, extended from our pure subtyping method, to determine the HIV-1 circulating recombinant forms (CRFs). That is, each of the 331 HIV-1 recombinant strains (196 CRF01AE, 52 CRF02AG, 3 CRF03AB, 3 CRF04CPX, 3 CRF05DF, 8 CRF06CPX, 7 CRF07BC, 4 CRF08BC, 5 CRF09CPX, 3 CRF10CD, 10 CRF11CPX, 10 CRF12BF, 6 CRF13CPX, 7 CRF14BG, 5 CRF1501B (AE/B), 2 CRF16A2D, 4 CRF18CPX, and 3 CRF19CPX) was partitioned into 50 equal parts, and there were 950 associated testings, for each of which the subtypes of the two closest reference sequences were reported. Then, similarly, for each of the distinct 13 subtypes, its occurrence frequency in these 1,900 predicted subtypes was calculated and every recombinant strain was represented as a 13-dimensional vector (for 7 randomly picked recombinant strains, their associated vectors are plotted in Figure 5.6 as the right 7 columns). The non-trivial portions of the predicted subtypes can be assigned as the recombinant form. For instance, for strain `AF179368`, 37.6% predicted subtypes are A1, 0.2% are F1, 2.6% are F2, and 59.6% are G. Therefore, we may predict that this strain is an A1G recombinant. The known recombinant form,

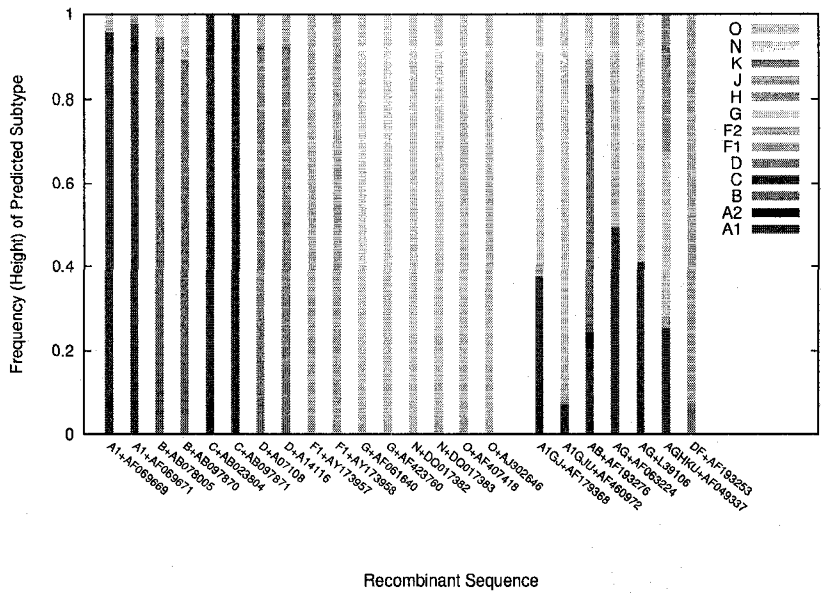


Figure 5.6: The frequencies of the predicted subtypes for the 16 of 825 randomly selected pure subtype strains and the 7 recombinants, using the 5,000 top ranked strings.

recorded in the LANL HIV-1 Sequence Database, is CRF11CPX and it is a mosaic of A/G/E/J. In other words, our computational prediction somehow missed the subtype J information. For each recombinant strain, we calculated the prediction accuracy as the percentage of correctly assigned subtypes among the 1,900 ones. The average prediction accuracy on those 91 recombinant strains (CRF02AG, CRF03AB, CRF05DF, CRF07BC, CRF08BC, CRF10CD, CRF12BF, CRF14BG, and CRF16A2D) which have the deterministic recombinant forms is 87.3%. Among the above 7 selected recombinant strains, our method perfectly predicted on `AF063224` and `L39106`, which are AG recombinants (Figure 5.6, the 20th and 21st columns).

It is claimed that the NCBI genotyping tool is "especially useful for the analysis of recombinant sequences" [70]. To consider this claim, we first conducted a comparative study by submitting all 331 recombinant strains to the server, but only allowed it to predict pure subtypes. For each sliding window, we reported the top two subtypes according to the BLAST similarity score, and similarly calculated the prediction accuracy. For the 91 strains having deterministic recombinant forms, the average prediction accuracy was 73.4%. Secondly, we replaced the 48 reference pure subtype strains in the tool by our 42 reference strains (in fact, our 42 are included in the 48) to test the BLAST methodology using the same set of reference strains. For the 91 strains having deterministic recombinant forms, the average prediction accuracy decreased a little to 66.2%. In another study, we allowed the server to report the closest recombinant form since it has reference recombinant strains. Among the collected 331 recombinant strains, 65 of them are used as references in the tool (the tool has in total 68 reference recombinant strains, in which 3 of them are absent from the LANL HIV-1 Sequence Database). For the other 266 recombinant strains, the server made only two mistakes where two CRF12BF strains, AY771588 and AY771589, were predicted to pure subtype B. The prediction results remained exactly the same even when the 48 reference pure subtype strains were replaced by our 42 reference strains. In the last study to test our pure subtyping method, we used our 42 reference pure subtype strains and the 68 reference recombinant strains in the NCBI tool, and the $5,000$ selected nucleotide strings to map every strain into a $5,000$-dimensional space and subsequently calculated all the pairwise distances. Each of the 266 testing recombinant strains was assigned the closest pure subtype or recombinant form. We were able to assign only 242 strains correctly, while all the other 24 strains were incorrectly assigned as CRF02AG. This indicates that the $5,000$ nucleotide strings are not good enough for recombinant form prediction, since they were selected for the purpose of pure subtyping. Nevertheless, it is interesting to see that both AY771588 and AY771589 were predicted to CRF02AG but not pure subtype B, suggesting that the $5,000$ selected nucleotide strings might capture some information missed by BLAST.

5.6 Conclusions

We proposed a method to select the most informative strings and use only their composition values to represent the whole genomes. Such a proposal appears novel in the context of HIV-1 subtyping and recombinant form determination. It reduces the genomic data dimensionality, and possibly reduces sequential evolutionary noise, and thus makes feasible the whole genome phylogenetic analysis on a large set of sequences. Such a method also enables us to identify informative explicit strings with respect to a large set of sequences and therefore supports biological explanation. Using our method to select 500 strings, for a total 867 pure subtype HIV-1 viral strains, we were able to predict their subtype perfectly, *i.e.*, 100% accuracy.

Chapter 6

Identifying Many Foot-and-Mouth Disease Virus Signature Nucleotide Strings for Computational Genotyping

6.1 Introduction

[1] Foot-and-mouth disease (FMD) is one of the most contagious animal diseases, with a large economic impact. Frequent sporadic outbreaks have been reported in many countries. The most recent outbreak in the United Kingdom costs tens of billions of dollars (`http://www.orau.gov/piadc/research.htm`). This disease causes extensive epidemics in domestic and wild cloven-hooved animals, such as cattle, sheep, goats, and pigs. In addition, it can result in persistent infections in hundreds of other animal species, and so the disease is in the importation banning and detection list of most countries. Thus, once this disease is identified, the infected animal populations are always required to be destroyed. The vaccination is the most efficient method to prevent this disease, so preparation and selection of an efficient vaccine will be the most important for FMD prevention and control.

FMD is caused by a single strand RNA virus, so-called FMD virus, which is a member of the family Picornaviridae, genus *Aphthovirus*. The genome of FMDV is about 8.4 kb in size, and encodes 12 proteins — leader proteinase L^{pro}, four structural proteins 1A (VP4), 1B (VP2), 1C (VP3), and 1D (VP1), and seven non-structural proteins 2A, 2B, 2C, 3A, 3B, 3C, and 3D. Like other RNA viruses, mutation and recombination always facilitate the emergence of new FMD strains. So far, seven immunologically distinct subtypes have been identified: Euroasiatic genotypes A, O, C, and Asia1 and South African Territories subtypes SAT1, SAT2, and SAT3. The capsid protein VP1 is exposed to the surface of the viron and contains genotype-specific amino acid sequence variations (the dissimilarity in nucleic acid among the genotypes can be up to 40%).

[1] This work has been published in BMC Bioinformatics [58].

Traditional laboratory experiments for subtyping were based on polymerase chain reaction (PCR) and nucleic acid hybridization. More recent methods, such as antigen capture RT/PCR (Ag-RT/PCR) [93], employ type-specific antibodies (against immuno-reactive recombinant proteins) for virus capturing followed by RNA amplification. These methods generally target only the VP1 gene, or VP1 and other capsid-coding genes. They are very useful for selecting the correct vaccines in case of FMD outbreak, but they may not be able to identify the presence of new variants or recombinants of multiple genotype viruses, which are very common cases for FMD. That identification is essential for determining the source of outbreak, understanding the evolution of the virus, and advancing the FMDV epidemiological study. Recently, advances in genomic sequencing technologies allow us to obtain rapidly the complete genomes of FMDV, and this enhances the development and application of computational strategies in genotype and genotype analyses of FMDV [30, 50, 93, 21]. Computational genotyping or genotyping analysis is generally based on a multiple sequence alignment of the viral genomic sequences or their protein products [50]. For example, the most recent FMDV phylogenetic and recombination analysis used split decomposition [30] to examine the complete strains of 103 isolates [21]. However, this strategy is limited by data size (particularly, the number of sequences), that is, the larger the dataset, the lower the accuracy that can be achieved.

In this study, we propose to identify a set of signature nucleotide strings which can be readily used to efficiently detect an emerging FMDV strain. Our method utilizes linear-kernel support vector machines as classifiers to extract the signature nucleotide strings using the complete composition vector (CCV) representation of the known FMDV strains [100]. In addition to genotype analysis, these signature strings may shed light on viral evolution, especially within the unique regions in the viral proteins; this information may be used for recombination vaccine construction.

We applied our method on a dataset of 129 FMDV whole genomes to predict the genotype for each of them; we achieved 98.45% leave-one-out cross validation (LOOCV) accuracy, only 2 were incorrectly assigned, due to virus recombination (see Discussion). An independent test on the other 37 strains achieved 100% accuracy using the selected nucleotide strings from the LOOCV study. More detailed analyses of the 20 top scoring strings revealed that these nucleotide strings are signature strings associated with genotypes. These results demonstrate that our proposal is promising in genotyping and further understanding evolutionary footprints of FMDVs.

6.2 Methods

6.2.1 Entropy-Based Nucleotide String Pre-Filtering

Using the CCV representation, each FMDV strain is mapped to an m-dimensional vector, where m could be as large as $4 + 4^2 + 4^3 + \ldots + 4^{15} = 1,431,721,300$ when setting the maximum string length to 15. Our computation on the dataset of 129 FMDV strains revealed that the number of present strings is $1,320,791$, about one thousandth of the largest possible number. Such a high-dimensional vector representation not only causes computational memory problem as demonstrated in our previous work [100], but also, and more severely, invokes the curse of dimensionality when genotyping is concerned. Disregarding the strain genotype information, among these nucleotide strings, many have very close (and small) composition values across all 129 strains, and thus they do not likely contain genotype-specific information. We design an entropy-based filter to retain only a number of nucleotide strings, which contain rich information content, for subsequent study.

Again, we use *relative entropy* to assign each strings a score and then keep only the top $10,000$ ranked strings (by their RREs). Two interesting facts about the FMDV dataset are that all these top ranked $10,000$ strings have length less than or equal to 11, and that 97.11% of them have lengths 6, 7, or 8. Such facts on one hand confirm partially the decision that we can skip longer strings in the analysis, on the other hand, support the observation that using a single string length, as is done in [40, 69], is not sufficient [100].

6.2.2 Disc-based Feature Selection Method

We combined the ACGS and DCGS feature selections into our method. Assume in the given multi-class whole dataset there are n strains on p strings, and these n strains belong to m subtypes. In the following, we define a novel vector representation for strains, which can differentiate their class recognition strength.

Let g_{ij} denote the CCV value of the i-th string in the j-th strain. That is, in the CCV matrix, each row represents a string, $G_i = g_{i1}, g_{i2}, \cdots, g_{in}$, and each column represents a strains. For the i-th string, its mean CCV value in the k-th subtype is denoted as h_{ik} for $k = 1, 2, \cdots, m$. The value $|h_{ik} - h_{il}|$ captures the difference between the mean CCV values of the i-th string in the k-th subtype and in the l-th subtype. Obviously, if this value is small, then the i-th string would not be effective in discriminating strains from these two subtypes, but it could be effective otherwise. Therefore, we define the subtype discrimination strength vector for the i-th string as

$$H_i = |h_{i1} - h_{i2}|, |h_{i1} - h_{i3}|, ..., |h_{i1} - h_{im}|, |h_{i2} - h_{i3}|, ..., |h_{i2} - h_{im}|, |h_{i3} - h_{i4}|, ..., |h_{i,m-1} - h_{im}|$$

Let $d_1(i_1, i_2)$ and $d_2(i_1, i_2)$ denote the Euclidean distances between the i_1-th and the i_2-th string based on their G-vectors and H-vectors (CCV values), respectively. Note that there are n entries in the G-vectors and $m(m-1)/2$ entries in the H-vectors, respectively. We define

$$d_{i_1,i_2} = \frac{d_1}{n} + \frac{2d_2}{m(m-1)}$$

to be the distance between the i_1-th and the i_2-th string. We can calculate the Euclidean distance by the above equation between every pair of strings, and then call the k-means algorithm to cluster the strings.

6.2.3 Genotype Signature String Extraction and an SVM-Classifier

With only $10,000$ nucleotide strings kept for analysis, the memory issue is resolved. For instance, the Euclidean distance between every pair of strains, using their $10,000$-dimensional vector representation, can be calculated. Such use of composition values treats all these $10,000$ nucleotide strings equally. A better way would be using these strings as features to classify the FMDV strains into different subtypes. However, the large gap between $10,000$ features and 129 FMDV strains could result in non-unique classifiers which would be significantly biased on these 129 strains. Our next step is to further select a smaller number of strings out of the $10,000$ to build an effective genotype predictor for novel strains. To do this, each strain is represented as a $10,001$-dimensional vector, in which the last entry records the genotype label. We then apply one of the most effective feature extraction methods, the Disc-F-test method [17]. Under the *leave-one-out cross validation* (LOOCV) scheme, Disc-F-test method first applies the F-test method [32] to re-order the $10,000$ strings, using their composition values in 128 of the 129 strains (called *training dataset*; the other strain is held out for testing purpose, whose genotype label is blinded to the constructed predictor). A string receives a high score if its composition values are close to each other in strains of the same genotype, but distinct to each other in strains of different genotypes. At the same time, the Disc-F-test method uses a k-mean algorithm to cluster the $10,000$ strings into 150 clusters, using again their composition values in all the 128 training strains, and additionally the differences between the mean composition values in different genotypes. The method then walks through the string order determined by the F-test method to pick up one string per cluster for the first 140 clusters. Note that setting up 10 more clusters in the k-mean clustering algorithm is to put away some strings that are not directly useful for genotype classification. These 140 selected strings, together with their composition values in all the 128 strains, are fed into a linear kernel SVM to build a classifier. Later on, for the testing strain, the composition values for only these 140 strings are calculated and such a 140-dimensional vector is sent to the SVM-classifier for genotype prediction.

6.3 Computational Results

The first FMDV dataset we collected contains in total 129 whole viral genomes, among which there are 47 genotype A, 48 genotype O, 8 genotype C, 9 genotype Asia1, 9 genotype SAT1, 4 genotype SAT2, and 4 genotype SAT3 strains (Table 6.1, second column). The average length of these whole genomic sequences is 8,151bp, with the maximum length being 8,280bp (with S fragment) and the minimum length being 6,996bp (without S fragment). These FMDV sequences were downloaded from GenBank at NCBI. The second dataset contains 37 strains, used for independent testing. Their genotype composition is recorded in the 10th column of Table 6.1.

Table 6.1: The composition of the different genotype FMDV strains in our two datasets (columns 2 and 10). The LOOCV prediction results on the first dataset are in columns 3–9; The independent testing results on the second dataset are in columns 11-17.

		A	O	C	Asia1	SAT1	SAT2	SAT3		A	O	C	Asia1	SAT1	SAT2	SAT3
	129	47	49	8	9	11	3	2	37	5	7	16	5	1	2	1
A	47	**47**							5	**5**						
O	48		**48**						7		**7**					
C	8			**8**					16			**16**				
Asia1	9				**9**				5				**5**			
SAT1	9					**9**			1					**1**		
SAT2	4		1				**3**		2						**2**	
SAT3	4					1		**2**	1							**1**

6.3.1 Baseline Clustering Results

Setting the maximum string length to 15, there are in total $1,320,791$ strings occurring in the first dataset of 129 FMDV strains. By computing the composition values for each of these strings and using the top $10,000$ ranked ones by RRE, every strain can be represented as a $10,000$-dimensional vector. Applying standard principle component analysis (PCA) on this $10,000 \times 129$ matrix to obtain the first two principle components (PCs), the linear discrimination analysis (LDA) shows that all the Euroasiatic genotype strains are well separated from those SAT strains, except a SAT1 strain `AY593844` and a SAT2 strain `AY593849` which appear close to Euroasiatic strains (Figure 6.1). Within Euroasiatic genotype strains, some of them, for example, some genotype O strains, show large distances from the other strains, while some others seem to mix together. Within SAT genotypes, the strains all mix together and are seemingly inseparable. Increasing the number of PCs (up to 9) can obtain some finer resolution results but the general conclusions remain the same (data not shown). These $10,000$ top ranked strings have their length in between 4 and 11 and the detailed percentages are collected in Table 6.2, where length-6, 7, 8 strings show dominant (97.11%).

Table 6.2: The percentages of different length strings in the top ranked 10,000 strings by their RREs.

String Length	4	5	6	7	8	9	10	11
Percentage (%)	0.01	0.59	12.27	57.43	27.41	2.11	0.12	0.06

Note that it is impossible to perform PCA on all $1,320,791$ strings. Nevertheless, we used the $1,320,791$-dimensional representation vectors to calculate the pairwise distances for all the 129 strains, and subsequently submitted them to the Neighbor-Joining method in Phylip 3.70 [33] to construct a phylogeny, shown in Figure 6.7. Clearly seen, though common genotype strains are largely clustered into separate clades, there are at least 7 misplacements in this tree. Applying the same procedure but using only the top ranked $10,000$ strings, the Neighbor-Joining tree shows at least 9 misplacements (Figure 6.8). Alternatively, we constructed a multiple sequence alignment for

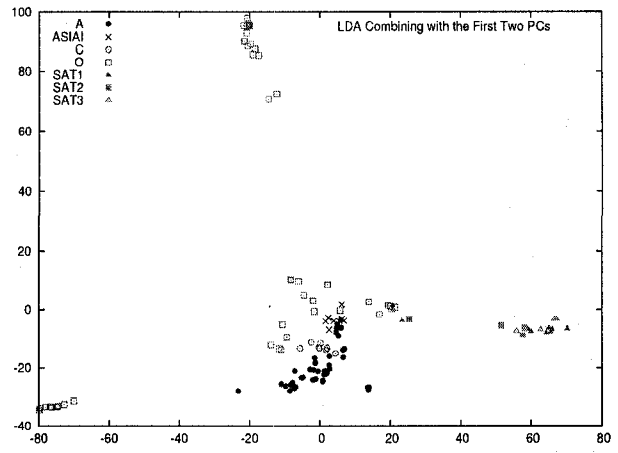


Figure 6.1: Two component LDA using the first two PCs from PCA on the 129 strains each represented as a $10,000$-dimensional vector.

the 129 strains, via ClusalW (which took 18.7 hours on a desktop with a 2.1GHz CPU and 2.0GB memory to complete), and the associated phylogeny shows a better result. Figure 6.9 shows the MSA tree in which at least the two strains, SAT1 strain `AY593844` and SAT2 strain `AY593849`, are misplaced.

6.3.2 RRE-LOOCV Genotyping Results

In the leave-one-out cross validation (LOOCV) scheme, at each iteration one strain is held out as testing sample while all the others, labeled by their genotypes, form a training dataset. Using all $1,320,791$ strings and the top ranked $10,000$ strings on the training dataset, respectively, we have calculated the euclidean distance from the testing strain to each of the 128 training strains, the average distance from the testing strain to each of the 7 genotypes, and finally assigned the testing strain with the closest genotype. We call this genotyping method the Mean-classifier, which has been adopted in many previous classification studies [90]. The LOOCV accuracy of this Mean-classifier is 123/129 = 95.35% and 108/129 = 84.72%, respectively. Using all $1,320,791$ strings and the top ranked $10,000$ strings, respectively, linear-kernel SVM classifiers were built and used to predict the genotype of the testing strain. The LOOCV accuracies of these two SVM-classifiers are both 120/129 = 93.02%.

Using only the top ranked k strings, for $k = 1,2,3,\ldots,140$, by RRE, we have also collected the LOOCV accuracies of the Mean-classifier and the SVM-classifier. These results are plotted in Figure 6.2. The highest accuracy by the Mean-classifier and the SVM-classifier were 111/129 = 86.05% and 118/129 = 91.47%, respectively. Figure 6.2 also shows the general tendency that the Mean-classifier performed slightly worse than the SVM-classifier.

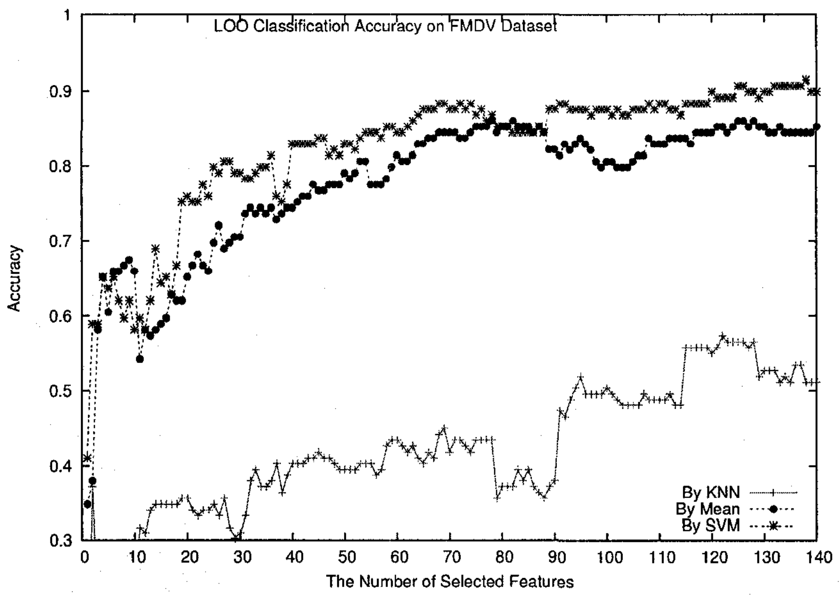


Figure 6.2: The LOOCV genotype prediction accuracies of the SVM-classifier and the Mean-Classifier using the top ranked strings by RRE.

6.3.3 Genotype Signature String Extraction and LOOCV Genotyping Results

RRE is expected to capture most of the information content carried by a string, disregarding the strain genotype information. Consequently, it might not differentiate well genotype signature strings from non-signature strings. The biomarker identification method, the Disc-F-test method, was employed to extract the most genotype-discriminative strings out of the $10,000$ (again, under LOOCV, selected on 128 training strains). Within this method, F-test is run to re-sort the $10,000$ strings, incorporating the strain genotype information. Using the top ranked k strings by the F-test method, for $k = 1,2,3,\ldots,140$, the LOOCV accuracy of the SVM-classifier is plotted in Figure 6.3 (labeled with F-test-SVM-classifier). Typically, when $k = 140$, the LOOCV accuracy reaches 123/129 = 95.35% (the highest by F-test-SVM-classifier). The LOOCV accuracy of the Mean-classifier using these k strings is also plotted in Figure 6.3 (labeled with F-test-Mean-classifier). Typically, when $k = 140$, the LOOCV accuracy reaches 124/129 = 96.12% (which is the highest for the F-test based classifiers).

The top ranked strings by the F-test method might have redundant genotype discriminatory power [17] and it is not beneficial to include all of them in building classifiers. The Disc-F-test method uses the k-means algorithm to cluster the $10,000$ strings into 150 clusters, using their composition values across the 128 training strains and the differences between the mean composition values in different genotypes (to define the euclidean distance), and then walks through the string order by the F-test method to pick up one string per cluster to build the SVM-classifier and the Mean-classifier. Using the first k strings by the Disc-F-test method, for $k = 1,2,3,\ldots,140$, the

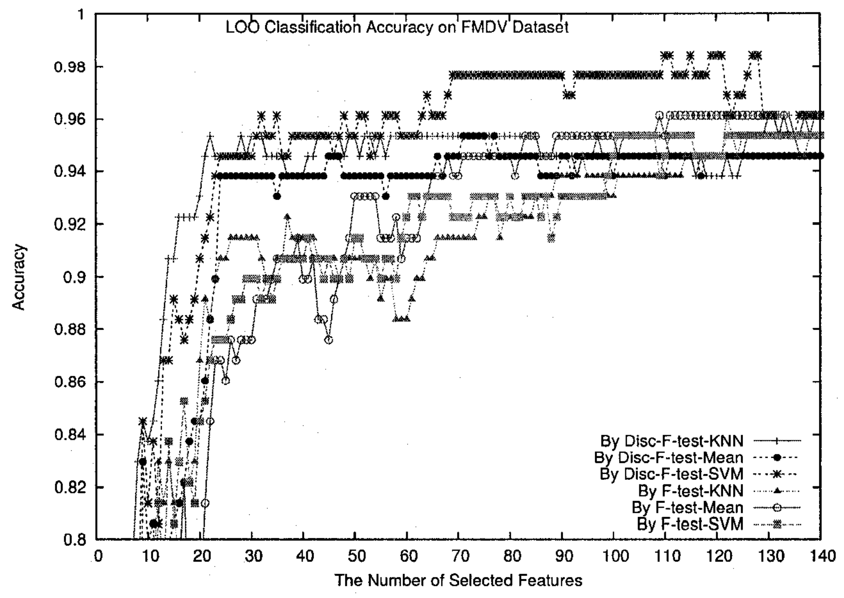


Figure 6.3: The LOOCV genotype prediction accuracies of the SVM-classifier and the Mean-classifier using the top ranked strings by the F-test method and the Disc-F-test method.

LOOCV accuracies of the SVM-classifier and the Mean-classifier are plotted in Figure 6.3 too (labeled with Disc-F-test-SVM-classifier and Disc-F-test-Mean-classifier, respectively). The highest LOOCV accuracy reached by the Mean-classifier is 123/129 = 95.35%, when using 71–75 and 77 selected strings; The highest LOOCV accuracy reached by the SVM-classifier is 127/129 = 98.45% (Table 6.1 columns 3–9, where SAT2 strain AY593849 was predicted as O and SAT3 strain AY593850 was predicted as SAT1), when using around 120 selected strings. Figure 6.3 shows a clear pattern that the Disc-F-test-SVM-classifier performed the best among the four.

For all the 129 testing sequences, their prediction confidence values of the Disc-F-test-Mean-classifier and F-test-Mean-classifier are plotted in Figure 6.4 , partitioned into different genotypes and in non-increasing order, where for the Disc-F-test-Mean-classifier only 8 of 129 predictions have confidence less than 0.1, which is the normal threshold for high confidence [90]. These 8 strains include 1 A, 2 C, 1 O, 1 SAT1, 2 SAT2, and 1 SAT3 strains.

6.3.4 Independent Genotyping Results

Using the first dataset of 129 FMDV strains as our training dataset, we applied the above procedure to firstly rank all the occurring strings by RRE, then re-rank the top 10, 000 of them by F-test using the strain genotype information (assuming they are all correct, see Discussion), and lastly select 140 strings using the Disc-F-test method. Notice that these 140 strings could slightly differ from each of the sets of 140 strings in the above LOOCV study. Using the linear-kernel SVM classifier built on them, every strain in the second dataset was submitted to have its genotype predicted. We achieved a 100% prediction accuracy in this independent test.

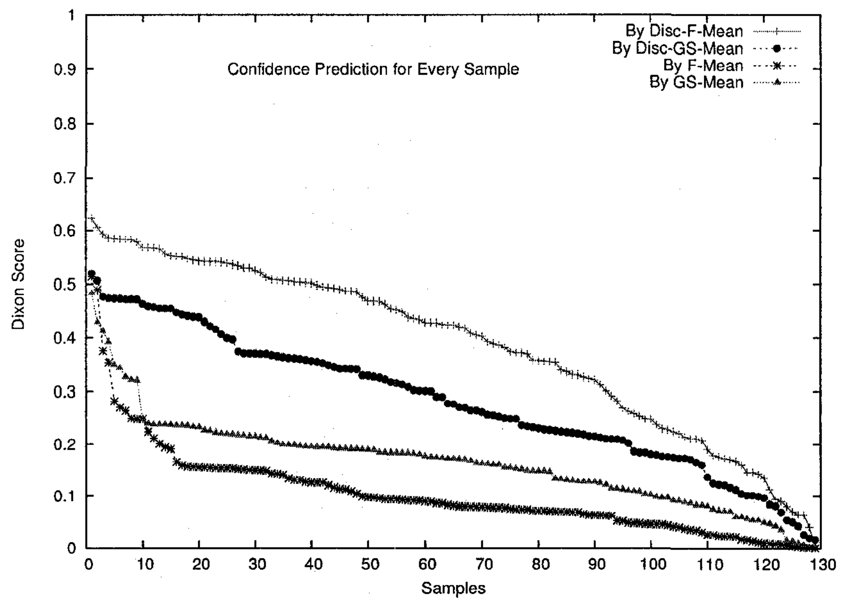


Figure 6.4: Prediction confidence values for the Disc-F-test-Mean-classifier

6.4 Discussion

6.4.1 Always Mis-Typed Strains

For each of the 129 viral genomes in the first dataset, we submitted it to BLAST and assigned its genotype using either the closest hit or a majority vote from 5NN. SAT1 strain AY593844 was incorrectly typed when using the closest hit, and four SAT2 strains AF540910, AY593847, AY593848, and AY593849 were mistyped using the second rule. We note that the genome database we used in BLAST contains more FMDV strains than we have, yet BLAST made some unexpected mistakes.

The linear-kernel SVM classifiers we constructed using 140 strings selected by RRE and Disc-F-test achieved 127/129 = 98.45% LOOCV genotyping accuracy and made mistakes on SAT2 strain AY593849 and SAT3 strain AY593850. One possible reason for that is the limited number of SAT isolates available, compared to Euroasiatic subtypes, which form an unbalanced training dataset. On the other hand, similar to many other RNA viruses, FMDVs have been reported with active recombination, which may be located in not only the structural protein coding regions but also the non-structural protein coding regions [21]. Phylogenetic analyses have shown incongruent topologies between the genes, for example, L^{pro}, $3C^{pro}$, and 1D [49, 73, 51, 98, 21]. Significantly, the above two mistyped genotypes were reported with potential recombination [21]. Strain AY593849 (SAT2/3 Kenya 11/60) and AY593850 (SAT3/2 SA57/59) were reported with conflicting phylogenetic topologies over the overall genomic sequences and different regions (*e.g.* L^{pro}/2A to 3D) [21]. We also examined these two mistyped strains by analyzing the different fragments on their genomes using our classifiers. The region-by-region analyses revealed that their P1 region, especially the VP1 gene which is used currently for the FMDV genotyping, is closely related to their Genbank recorded genotype. However, the other different regions in their genomes are closer to the genotypes predicted by our classifiers than to their recorded genotypes. In fact, the top 10 strings we used for genotyping are all outside of the VP1 region (Table 6.3), which further supports our prediction methods. For example strain AY593850: its VP2, VP3 and VP1 genes showed high similarities in sequence with isolates of SAT3. While in P2, P3 and 3' UTR regions, it shares more sequences with those of SAT1 and SAT2, rather than SAT3. For strain AY593849, a BLAST search with its full sequence, the top 5 isolates which have statistically significant sequence matches are one SAT2, one SAT1, and three Asia1. Therefore we believe that the mistyped results resulted from potential recombination between FMDVs. Future study will be to identify the potential recombination cases using signature string information.

6.4.2 The Maximum String Length

In our experiments, we set the maximum string length to 15. The rationale on setting this value is that increasing it does not improve the genotyping accuracy, since essentially strings of length greater than 11 do not make it into the list of top ranked 10,000 strings by RRE. Nevertheless, long genotype-specific amino acid motifs have been discovered, for example, YSTXEDHXXGPN is genotype A specific [21]. We have therefore increased the maximum string length to 26[2] and applied all the above the same methods to perform the genotyping. Once again, we found no improvement, and only 2 length-16 strings were able to make into the $10,000$ strings yet still not selected by the F-test or Disc-F-test methods (data not shown).

6.5 More Information on the Top Strings Selected by Disc-F-test

We also examined how well the top 20 strings selected by the Disc-F-test method cover the positions in the whole genomes. For this purpose, we constructed in the multiple alignment of these 129 strains by ClustalW (which took 18.7 hours on a desktop with a 2.1GHz CPU and 2.0GB memory to complete), and for each of the 20 selected strings, we found all of its occurrences and highlighted them in the multiple sequence alignment. The top ranked string CCGCCTG appears twice in most of the Euroasiatic strains (80/112, MSA locations 794 and 4,349), twice in all SAT3 strains (MSA locations 4,349 and 7,097), but only once in SAT1 and SAT2 strains (12/13, MSA location 4,349), and therefore its abundance might be regarded as a signature for distinguishing the Euroasiatic strains and the SAT strains; In addition, string CCGCCTG appears in 107/112 Euroasiatic strains at MSA location 794, but in none of the SAT strains; it appears in all SAT3 strains at MSA location 7,097, which could be SAT3 specific — both indications of its distinguishing role; The second ranked string TAAGGTA appears in only one Euroasiatic strain, but in 15 of the 17 SAT strains (MSA lo-

[2]The experiment failed on the maximum length 27 due to insufficient memory. All these experiments were done on a Heisler cluster node with a 2.2Ghz CPU and 5.0GB memory.

cation 1,020), and therefore it can be regarded as a signature string for the SAT strains; The third ranked string `AGTCCAT` appears in none of the Euroasiatic strains, but in 16 of the 17 SAT strains (MSA location 8,008), and therefore it can also be regarded as a signature string for the SAT strains; The fourth ranked string `TTCATCAA` appears in most of the non-A strains (71 out of 82), but only in one of the 47 genotype A strains (MSA locations 1,647 and 2,520), indicating its unlikeliness in genotype A strains; The fifth ranked string `ACCGACGG` appears in four genotype O strains (MSA location 92) and 15 of the 17 SAT strains (MSA location 5,278); The sixth ranked string `CCAGTGAA` appears in MSA location 6,177, in only 1 genotype A strain, but in all the 17 SAT strains; The seventh ranked string `GCGACAAC` appears mostly in MSA location 1,942, and in more than 50% of SAT strains; At MSA location 2,536, the eighth ranked string `ACCAACAT` appears in 29 genotype A strains (62%), in all the 9 Asia1 strains (100%), in half of the C strains, and once in a SAT1 strain, but not the others; The ninth ranked string `GTTTCT` appears in several locations in the MSA, but its major occurrences are at location 1,046, where it appears in 15 out of the 17 SAT strains (but no Euroasiatic strains), and at location 4,561, where it appears in 88%, 89%, 87%, and 92% genotype A, Asia1, C, and O strains, respectively (but only 1 SAT1 and 1 SAT2 strains); Similarly, the major occurrences of the tenth ranked string `CACATGG` is at MSA location 8,046, where it appears in 89%, 89%, 100%, and 100% of genotype A, Asia1, C, and O strains respectively, while in only 1 SAT2 strain.

Table 6.3: Summary of the top 10 strings in the MSA.

Rank	String	MSA Location	genotype	Gene / Region
1	`CCGCCTG`	794	A, Asia1, C, O	between polyC and L^{pro}
		7,097	SAT3	3D: RNA polymerase
2	`TAAGGTA`	1,020	SAT1-3	between polyC and L^{pro}
3	`AGTCCAT`	8,008	SAT1-3	3D: RNA polymerase
4	`TTCATCAA`	1,647, 2,520	Asia1, C, O	L^{pro} and VP2 (1B)
5	`ACCGACGG`	5,278	SAT1-3	2C
6	`CCAGTGAA`	6,177	SAT1-3	3B
7	`GCGACAAC`	1,942	SAT1-3	VP4 (1A)
8	`ACCAACAT`	2,536	A, Asia1, C	3' end of VP2 (1B)
9	`GTTTCT`	1,046	SAT1-3	between polyC and L^{pro}
		4,561	A, Asia1, C, O	2B
10	`CACATGG`	8,046	A, Asia1, C, O	3D: RNA polymerase

In summary, among these top 20 strings, `ACCGACGG`, `AGTCCAT`, `CCAGTGAA`, `CGCTCCAA`, `TAAGGTA` seem to be SAT specific, `ACGCGA`, `CACATGG`, `CACGGTC` seem to be Euroasiatic specific, and the occurrences at different locations of `GTGTTTGA` and `GTTTCT` seem to distinguish Euroasiatic strains from SAT strains; Moreover, string `ACCAACAT` recognizes non-O Euroasiatic strains, string `AGCTGACC` is only abundant in genotype A strains (91%), string `TCACGGT` seems to recognize genotype A and SAT strains, and string `TTCATCAA` recognizes non-A strains. By retaining only

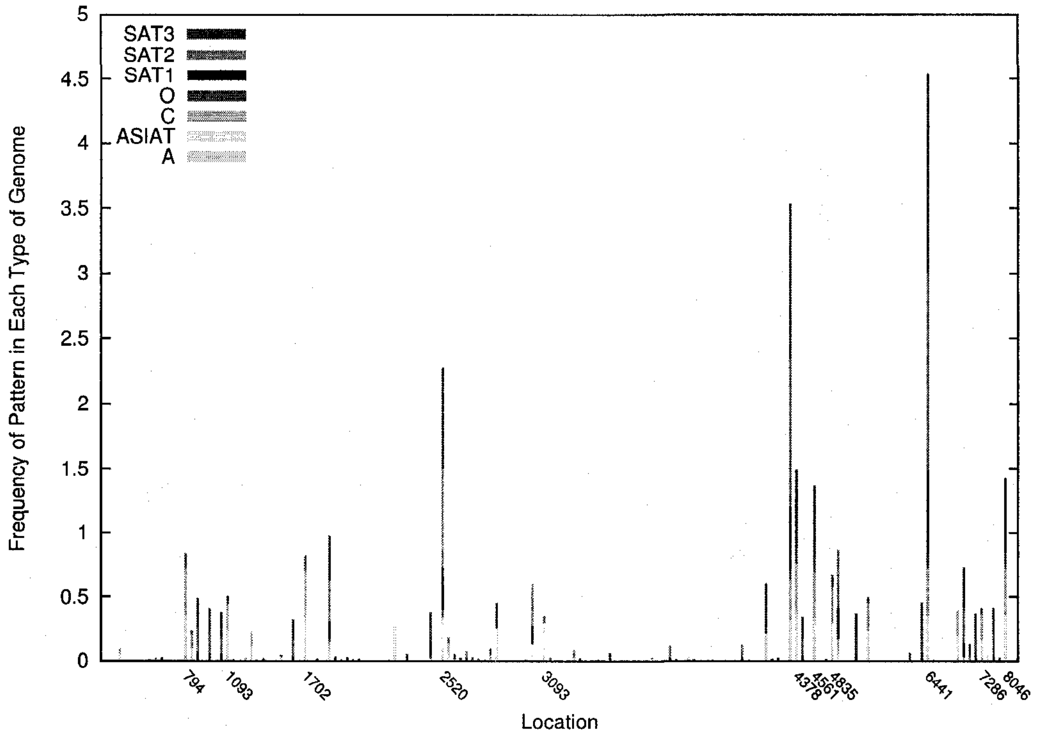


Figure 6.5: The average occurrence frequency of the top 20 strings, by the Disc-F-test method, in each of the seven genotypes at ClustalW's MSA locations, among all the 129 whole genomes.

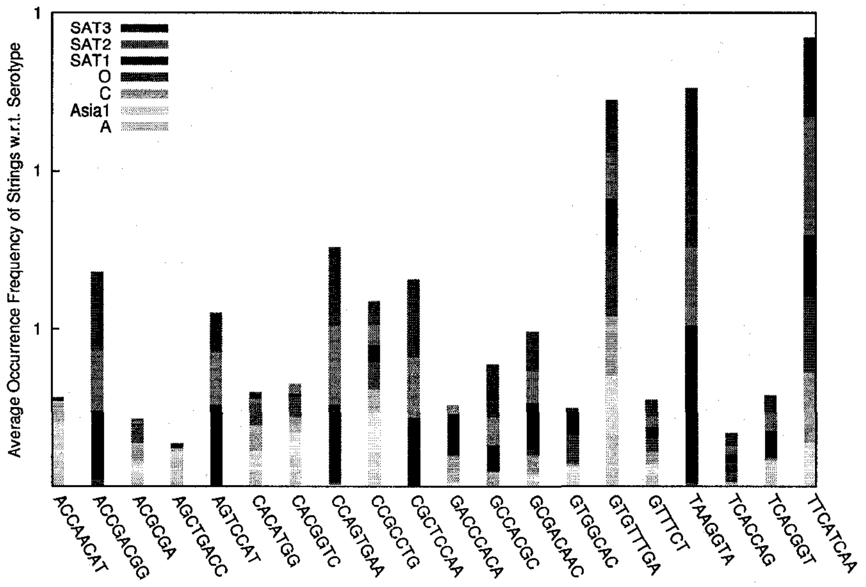


Figure 6.6: The average occurrence frequency of the top 20 strings, by the Disc-F-test method, in each of the seven genotypes, among all the 129 whole genomes.

those columns where at least one of the 20 strings occurs in the multiple sequence alignment, we calculated at each of them the average occurrence frequency of the strings with respect to the seven genotypes, which is shown in Figure 6.5. We have also calculated, for each of the top 20 strings, its average occurrence frequency with respect to the seven genotypes, and plotted them in Figure 6.6. These two plots clearly show that some of these top 20 strings do distinguish genotypes and some of their occurring locations do map to previously known genotype-specific genome regions such as the genotype-specific region in 1B and C-terminal of 1D [21]. As shown in Table 6.3, the 5' terminus of the genomic long fragment has three from these top 10 strings. This region may be important in cap-independent translation initiation of the viral polyprotein as well as genome replication [26]. The other three important strings are from 3D: RNA polymerase. Therefore these two regions are important in genomic replication and polyprotein translation. These may suggest the important roles of these genomic regions during viral adaptation to different environmental factors.

6.5.1 The Number of Pre-Selected Strings

We have also experimented with the top 20,000 ranked strings by their relative entropies. These 20,000 top ranked strings have their length in between 4 and 15 and the detailed percentages are collected in Table 6.4, where clearly seen that length-6, 7, 8 strings are still dominant (93.445%). There are some longer strings included compared to the top 10,000 strings, but there are only 0.02% or 4 such strings.

Table 6.4: The percentages of different length strings in the top ranked 20,000 strings by their relative entropies.

String Length	4	5	6	7	8	9	10	11	12	13	15
Percentage	0.02	0.86	11.875	47.475	34.105	5.225	0.345	0.075	0.01	0.005	0.005

Surprisingly, applying the same string selection methods coupled with the SVM- and Mean-classifiers (built on up to 140 strings) did not improve the serotyping accuracy, but decreased a little. The highest accuracy observed is 125/129 = 96.90%, by Disc-F-test-SVM-classifier, when selecting 100 strings (Figure 6.10).

The set of the top 20 strings shares 10 strings with the set of the top 20 strings selected using the same procedure on the preselected 10,000 strings, whose ranks are 1–5 and 7–11. This fact also shows that these 10 strings carry rich serotype specific information.

6.6 Conclusions

We proposed a method to select the most informative and genotype-specific composition nucleotide strings for FMDV whole genome genotyping. Such a proposal appears novel in the context of FMDV genotyping, with at least three advantages: 1) It does not involve the highly complex stage of multiple sequence alignments, and thus supports high throughput genotyping. This simplifies the genotyping process through sequencing information and thus shortens the disease control process — once the genotype is defined, the decision on the type of vaccine can be made. Therefore, it helps to determine the source of FMDV in case of an outbreak. The potential ability to recognize a recombinant, in addition to genotype FMDV, makes our method very valuable, especially for the war against bio-terrorism. 2) It considers the whole genomic sequence for genotyping, and filters potential random mutation at the same time. Therefore, it provides an additional and/or complementary genotyping tool to the current available methods. 3) It adopts feature selection methods to identify composition strings that are the most genotype-specific, and thus allows biological explanation on the genotyping results. The identified signature strings may also facilitate the preparation of recombination vaccine. It is interesting, but not completely unexpected, to see that using only around 120 strings selected by the Disc-F-test method, the genotyping accuracy on the set of 129 FMDV whole genomes by the SVM-classifier can reach as high as 98.45% (this is never achieved previously). Moreover, detailed examination of the top strings by the Disc-F-test method reveals that they are biologically meaningful, in that each of them either serves as a signature string for distinguishing some genotypes from the others, and/or maps well to previously discovered genotype-specific RNA/peptide motifs.

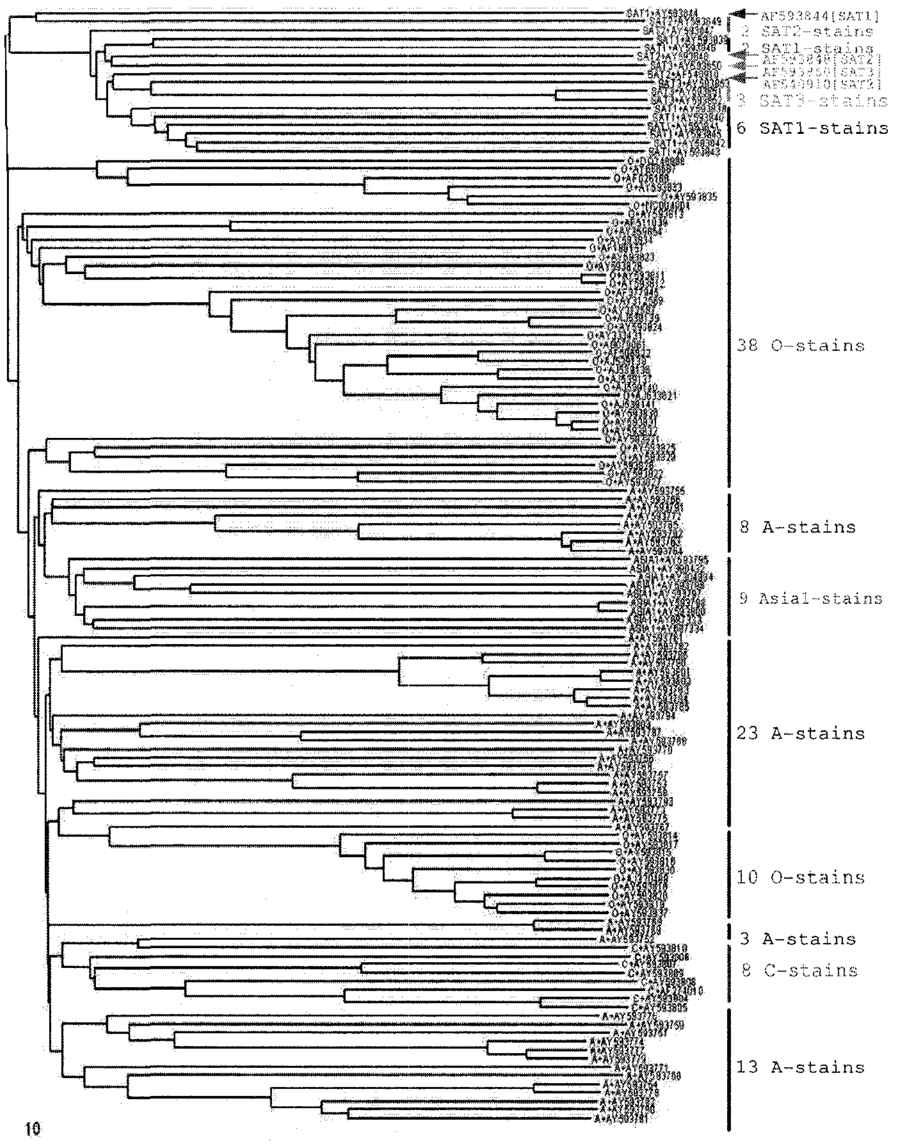


Figure 6.7: The Neighbor-Joining tree on the 129 FMDV strains each represented as a 1, 320, 791-dimensional vector. In this tree, again, Euroasiatic strains and SAT strains are well separated, but they internally seem to mix up.

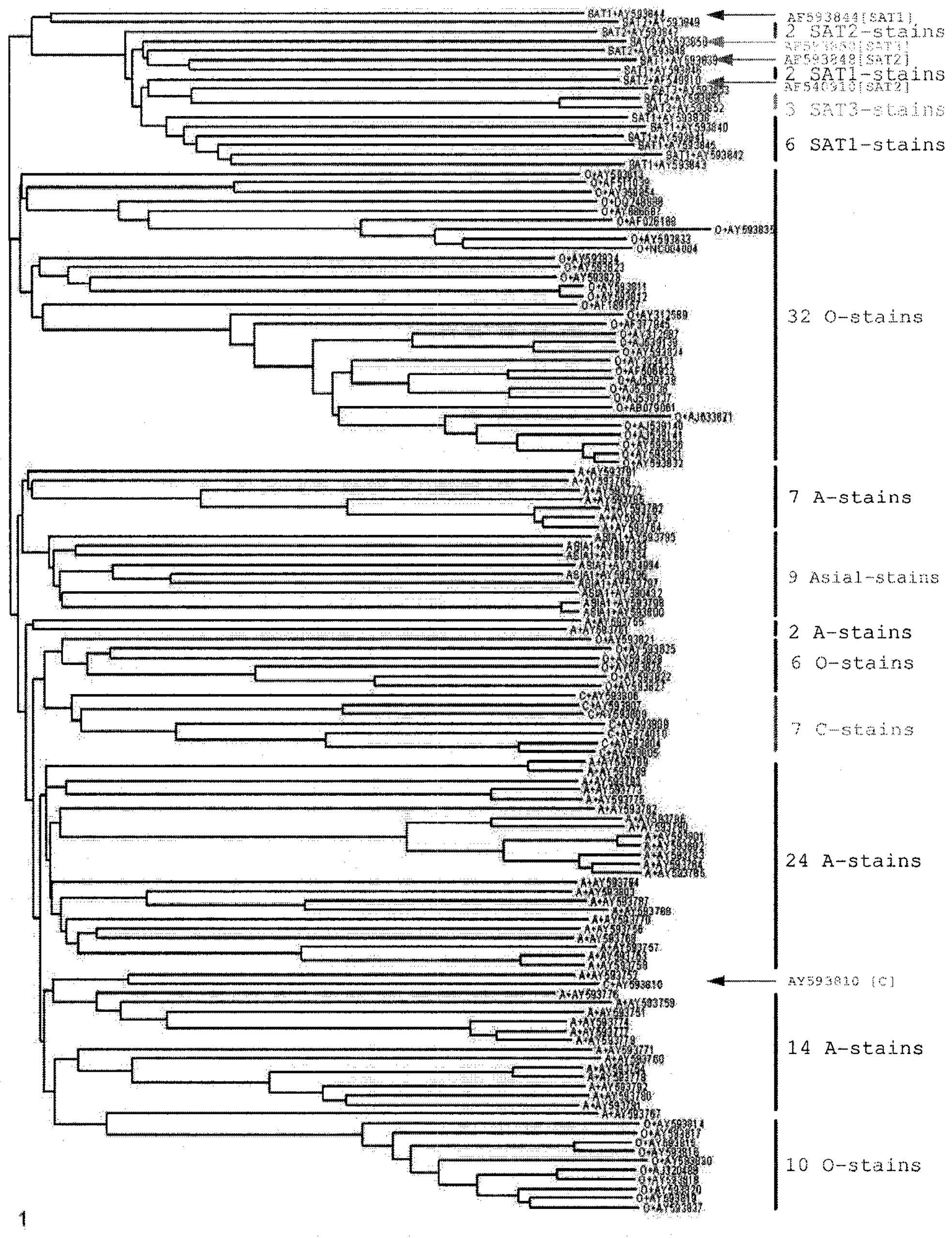


Figure 6.8: The Neighbor-Joining tree on the 129 FMDV strains each represented as a 10,000-dimensional vector. In this tree, again, Euroasiatic strains and SAT strains are well separated, but they internally seem to mix up.

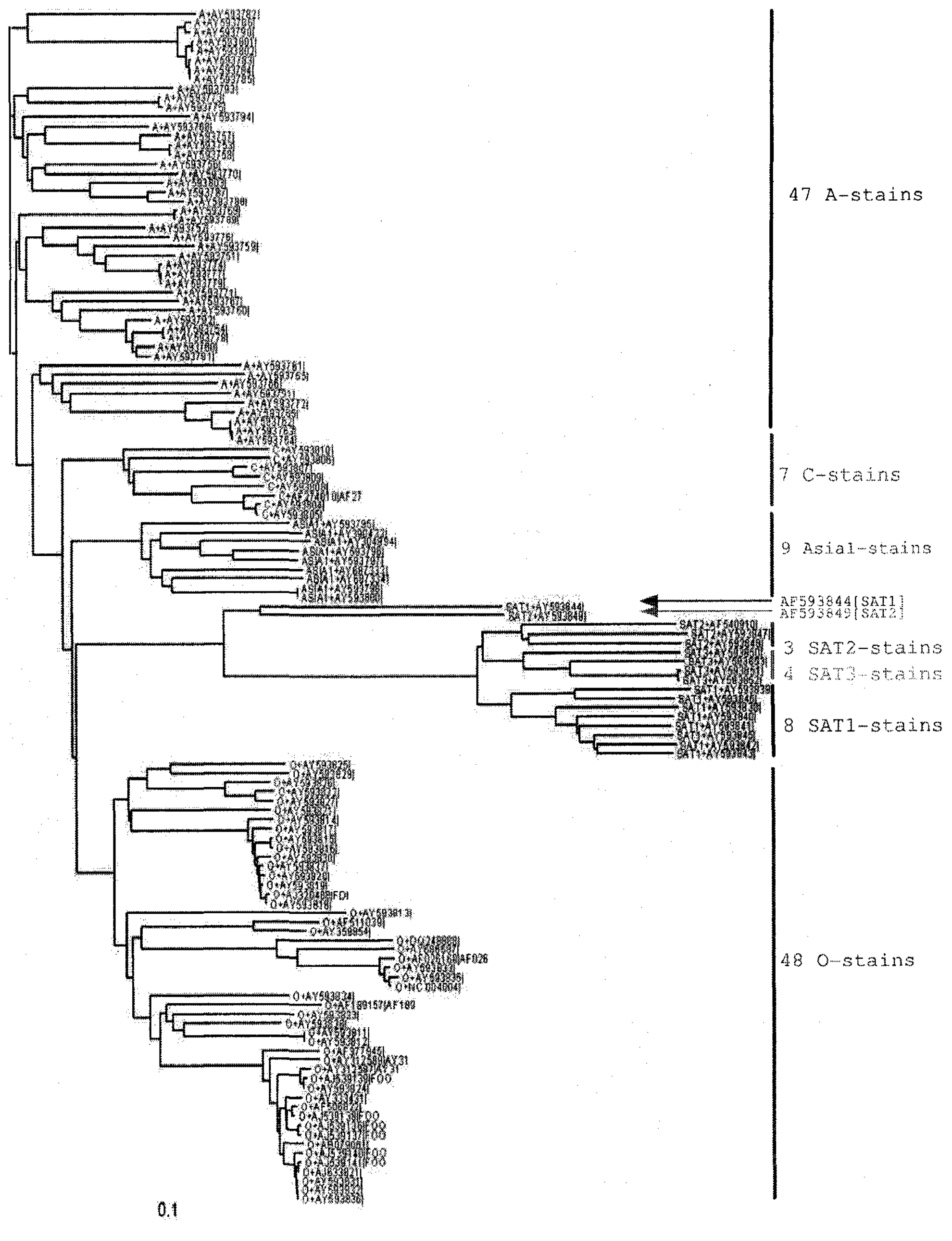


Figure 6.9: The MSA tree on the 129 FMDV strains using their whole viral genomes. In this tree, only two SAT strains seem to be misplaced.

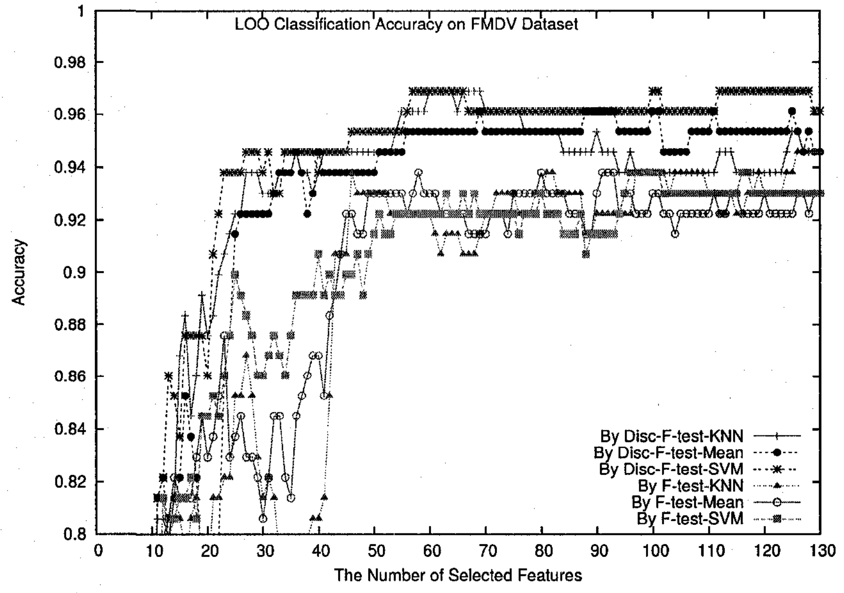


Figure 6.10: The LOOCV genotype prediction accuracies of the SVM-classifier and the Mean-classifier using the strings selected by the F-test method and the Disc-F-test method, from the 20,000 strings.

Chapter 7

Conclusions

The ultimate goals of bioinformatic studies are to extract information through the large volume of data. Bioinformatic data varys in their presentation and specifications. At current stage, it is difficult to propose a universal instrument for different types of data. In this thesis, we investigated the characteristics of two types of data, gene expression microarray data and viral whole genome data, and proposed several effective algorithms for them. Our methods are based on the concept of feature selection. For gene expression microarray data, we selected feature genes and built classifiers. We also proposed missing value estimation methods to tackle the common problem of missing data in microarray. The application of our methods on cancer datasets showed desirable performance. For viral whole genome data, we selected feature nucleotide or amino acid and built efficient serotypers. The algorithms showed success on different types of viral data. From all the experimental results, we believe that our methods are efficient and suitable tools for gene expression microarray and viral whole genome data mining. Because of the specificity of different types of data, we are uncertain whether our methods can be applied to other bioinformatic data. However, we think that the feature selection principle is essential to solve the problem of data complexity inherent in bioinformatics, and should be applicable to other kinds of data.

Appendix

During my PH. D study, I have also been involved in several theoretic problems.

Capacitated Multicast Tree Routing Problem

The Capacitated Multicast Tree Routing Problem is considered, in which only a limited number of destination nodes are allowed to receive data in one routing tree and multiple routing trees are needed to send data from the source node to all destination nodes. The goal is to minimize the total cost of these routing trees. The problem is NP hard. The first approximation algorithm proposed by us has ratio $2 + \rho$, where ρ denotes the best approximation ratio for the Steiner Minimum Tree problem, and it is about 1.55 at the writing of the thesis [1]. The ratio was further improved to $\frac{8}{5} + \frac{5}{4}\rho$ which is still the current best ratio for this problem [2].

Path Covering on Trees with its Applications in Machine Translation

Given a tree and a set of paths in the tree, the problem of finding a minimum number of paths from the given path set to cover all the vertices in the tree is investigated in the paper. To distinguish from the classical path cover problem, such an optimization problem is referred to as vertex covering by paths. The problem and its edge variant, edge covering by paths, find applications in machine translation. We have shown that the problem of finding a minimum number of given paths in a tree to cover all the vertices (all the edges) in the tree is hard. Such a combinatorial optimization problem arises from an application in machine translation. We have developed an exact recursive algorithm for solving the general problem, which runs in an exponential time in the worst case but becomes a polynomial time algorithm when every vertex in the tree is included in a bounded number of given paths. This special case maps to the machine translation application. To summarize, we answered an open question posed in [56] on the computational complexity of the VcpT problem. We have also presented a 2-approximation algorithm for the VcpT problem [3].

[1]This work has been published in Proceedings of the 11th International Computing and Combinatorics Conference [19].

[2]This work has been published in the 2nd Annual International Conference on Combinatorial Optimization and Applications [15].

[3]This work has been published in Information Processing Letters [57].

Remark

- The contents of the Chapter 2 are based on the paper published in Journal of Bioinformatics and Computational Biology [18].
- The contents of Chapter 3 are based on the two papers published in BMC Bioinformatics [16, 102].
- The contents of Chapter 4 are based on the papers published in EMBC [20].
- The contents of Chapter 5 are based on the paper published in Bioinformatics [99].
- The contents of Chapter 6 are based on the paper published in BMC Bioinformatics [58].

Bibliography

[1] http://employees.csbsju.edu/hjakubowski/classes/ch331/bind/microarray.gif.

[2] http://www.thatsfit.com/2007/12/17/.

[3] http://www.cancerbackup.org.uk/Aboutcancer/Whatiscancer/Whatiscancer.

[4] http://en.wikipedia.org/wiki/DNA_microarray.

[5] http://www.cs.wustl.edu/~jbuhler/research/array.

[6] http://www.affymetrix.com/index.affx.

[7] S. F. Altschul, T. L. Madden abd A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[8] S. A. Armstrong, J. E. Staunton, L. B. Silverman, R. Pieters, M. L. den Boer, M. D. Minden, S. E. Sallan, E. S. Lander, T. R. Golub, and S. J. Korsmeyer. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, 30:41–46, 2002.

[9] P. Baldi and A. D. Long. A Bayesian framework for the analysis of microarray expression data: Regularized t-test and statistical inferences of gene changes. *Bioinformatics*, 17:509–519, 2001.

[10] M. Barenco, J. Stark, D. Brewer, D. Tomescu, R. Callard, and M. Hubank. Correction of scaling mismatches in oligonucleotide microarray data. *BMC Bioinformatic*, 7:251, 2006.

[11] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, Javad Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. J. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerso. Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of National Academy of Sciences of the United States of America*, 98:13790–13795, 2001.

[12] F. Blanchot-Jossic, A. Jarry, D. Masson, K. Bach-Ngohou, J. Paineau, M. G. Denis, C. L. Laboisse, and J. F. Mosnier. Up-regulated expression of ADAM17 in human colon carcinoma: co-expression with EGFR in neoplastic and endothelial cells. *Oncogene*, 207:156–163, 2005.

[13] T. H. Bø, B. Dysvik, and I. Jonassen. LSimpute: accurate estimation of missing values in microarray data with least squares methods. *Nucleic Acids Research*, 32:e34, 2004.

[14] A. J. Butte, J. Ye, G. Niederfellner, K. Rett, H. Hring, M. F. White, and K. P. White. Determining significant fold differences in gene expression analysis. *Pacific Symposium on Biocomputing*, 6:6–17, 2001.

[15] Z. Cai, Z. Chen, G. Lin, and L. Wang. An improved approximation algorithm for the capacitated multicast tree routing problem. In *the 2nd Annual International Conference on Combinatorial Optimization and Applications*, 2008.

[16] Z. Cai, R. Goebel, M. Salavatipour, Y. Shi, L. Xu, and G. Lin. Selecting genes with dissimilar discrimination strength for class prediction. In *Proceedings of The Fifth Asia-Pacific Bioinformatics Conference (APBC 2007)*, pages 81–90, 2007.

[17] Z. Cai, R. Goebel, M. R. Salavatipour, and G. Lin. Selecting dissimilar genes for multi-class classification: an application in cancer subtyping. *BMC Bioinformatics*, 8:206, 2007.

[18] Z. Cai, M. Heydari, and G. Lin. Iterated local least squares microarray missing value imputation. *Journal of Bioinformatics and Computational Biology*, 4(5):935–957, 2006.

[19] Z. Cai, G. Lin, and G. Xue. Improved approximation algorithms for the capacitated multicast routing problem. In *Proceedings of The 11th International Computing and Combinatorics Conference (COCOON 2005)*, 2005.

[20] Z. Cai, Y. Shi, M. Song, R. Goebel, and G. Lin. Smoothing blemished gene expression microarray data via missing value imputation. In *Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2008.

[21] C. Carrillo, E. R. Tulman, G. Delhon, Z. Lu, A. Carreno, A. Vagnozzi, G. F. Kutish, and D. L. Rock. Comparative genomics of foot-and-mouth disease virus. *Journal of Virology*, 79:6487–6504, 2005.

[22] H. Chai and C. Domeniconi. An evaluation of gene selection methods for multi-class microarray data classification. In *In Proceedings of the Second European Workshop on Data Mining and Text Mining for Bioinformatics*, pages 7–14, 2004.

[23] X. Chen, S. Kwong, and M. Li. A compression algorithm for dna sequences and its applications in genome comparison. In *In Proceedings of the Sixth Annual International Computing and Combinatorics Conference (RECOMB)*, pages 107–117, 2000.

[24] J. Cho, D. Lee, J. H. Park, and I. B. Lee. New gene selection for classification of cancer subtype considering within-class variation. *FEBS Letters*, 551:3–7, 2003.

[25] T. de Oliveira, K. Deforche, S. Cassol, M. Salminen, D. Paraskevis, C. Seebregts, J. Snoeck, E. J. van Rensburg, A. M. J. Wensing, D. A. van de Vijver, C. A. Boucher, R. Camacho, and A.-M. Vandamme. An automated genotyping system for analysis of HIV-1 and other microbial sequences. *Bioinformatics*, 21:3797–3800, 2005.

[26] S. Lopez de Quinto and E. Martinez-Salas. Conserved structural motifs located in distal loops of aphthovirus internal ribosome entry site domain 3 are required for internal initiation of translation. *Journal of Virology*, 71:4171–4175, 1997.

[27] R. Diaz-Uriarte and S. Alvarez de Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7, 2006.

[28] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *Proceedings of IEEE Computer Society Bioinformatics Conference (CSB03)*, pages 523–530, 2003.

[29] H. Dopazo, J. Santoyo, and J. Dopazo. Phylogenomics and the number of characters required for obtaining an accurate phylogeny of eukaryote model species. *Bioinformatics*, 21:i116–i121, 2004.

[30] J. Dopazo, A. Dress, and A. V. Haeseler. Split decomposition: A technique to analyze viral evolution. *Proceedings of National Academy of Sciences USA*, 90:10320–10324, 1993.

[31] S. Dreiseitl and L. Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35:352–359, 2002.

[32] S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97:77–87, 2002.

[33] J. Felsenstein. *PHYLIP*. `http://evolution.genetics.washington.edu/phylip.html`.

[34] A. P. Gasch, M. Huang, S. Metzner, D. Botstein, S. J. Elledge, and P. O. Brown. Genomic expression responses to DNA-damaging agents and the regulatory role of the yeast ATR homolog meclp. *Molecular Biology of the Cell*, 12:2987–3003, 2001.

[35] G. C. Gawley and N. Talbot. Gene selection in cancer classification using sparse logistic regression with Bayesian regularisation. *Bioinformatics*, 22:2348–2355, 2006.

[36] R. Gifford, T. de Oliveira, A. Rambaut, R. E. Myers, C. V. Gale, D. Dunn, R. Shafer, A. M. Vandamme, P. Kellam, and D. Pillay. Assessment of automated genotyping protocols as tools for surveillance of HIV-1 genetic diversity. *AIDS*, 20:1521–1529, 2006.

[37] 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, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[38] S. Grumbach and F. Tahi. A new challenge for compression algorithms: genetic sequences. *Information Processing and Management: an International Journal*, 30:866–875, 1994.

[39] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[40] B. Hao and J. Qi. Prokaryote phylogeny without sequence alignment: from avoidance signature to composition distance. In *In Proceedings of the 2003 IEEE Bioinformatics Conference (CSB 2003)*, pages 375–385, 2003.

[41] E. A. Herniou, T. Luque, X. Chen, J. M. Vlak, D. Winstanley, J. S. Cory, and D. R. O'Reilly. Use of whole genome sequence data to infer baculovirus phylogeny. *Journal of Virology*, 75:8117–8126, 2001.

[42] C. H. House and S. T. Fitz-Gibbon. Using homolog groups to create a whole-genomic tree of free-living organisms: an update. *Journal of Molecular Evolution*, 54:539–547, 2002.

[43] R. A. Irizarry, B. M. Bolstad, F. Collin, L. M. Cope, B. Hobbs, and T. P. Speed. Summaries of affymetrix genechip probe level data. *Nucleic Acids Research*, 31:4e15, 2003.

[44] R. Jörnsten, H.-Y. Wang, W. J. Welsh, and M. Ouyang. DNA microarray data imputation and significance analysis of differential expression. *Bioinformatics*, 21:4155–4161, 2005.

[45] S. Karlin and C. Burge. Dinucleotide relative abundance extremes: a genomic signature. *Trends in Genetics*, 11:283–290, 1995.

[46] L. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, and P. S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7:673–679, 2001.

[47] H. Kim, G. H. Golub, and H. Park. Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics*, 20:1–12, 2005.

[48] K. Y. Kim, B. J. Kim, and G. S. Yi. Reuse of imputed data in microarry analysis increases imputation efficiency. *BMC Bioinformatics*, 5:160–169, 2004.

[49] A. M. Q. King, D. McCahon, and K. Saunders. Molecular relationships between type Asia 1 new strain from china and type O panasia strains of foot-and-mouth-disease virus. *Virus Genes*, 35:273–279, 2007.

[50] N. J. Knowles and A. R. Samuel. Molecular epidemiology of foot-and-mouth disease virus. *Virus Research*, 91:65–80, 2003.

[51] O. Krebs and O. Marquardt. Identification and characterization of foot-and-mouth disease virus O1 burgwedel/1987 as an intertypic recombinant. *Journal of Genetic Virology*, 73:613–619, 1992.

[52] K. E. Lee, N. Sha, E. R. Dougherty, M. Vannucci, and B. K. Mallick. Gene selection: a Bayesian variable selection approach. *Bioinformatics*, 19:90–97, 2003.

[53] T. Leitner, B. Korber, M. Daniels, C. Calef, and B. Foley. *HIV-1 subtype and circulating recombinant form (CRF) reference sequences.* Accessible through `http://www.hiv.lanl.gov/content/hiv-db/REVIEWS/RefSeqs2005/RefSeqs05.html/`.

[54] L. Li, C. R. Weinberg, T. A. Darden, and L. G. Pedersen. Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method. *Bioinformatics*, 17:1131–1142, 2001.

[55] W. Li, W. Fang, L. Ling, J. Wang, Z. Xuan, and R. Chen. Phylogeny based on whole genome as inferred from complete information set analysis. *Journal of Biological Physics*, 28:439–447, 2002.

[56] D. Lin. A path-based transfer model for machine translation. In *Proceedings of The 20th International Conference on Computational Linguistics*, pages 625–630, 2004.

[57] G. Lin, Z. Cai, and D. Lin. Path covering on trees with its applications in machine translation. *Information Processing Letters*, 97, 2005.

[58] G. Lin, Z. Cai, J. Wu, X-F. Wan, L. Xu, and R. Goebel. Identifying a few foot-and-mouth disease virus signature nucleotide strings for computational genotyping. *BMC Bioinformatics*, 9:279, 2008.

[59] H. Liu and H. Motoda. Feature selection for knowledge discovery and data mining. *Boston, Kluwer Academic Publishers*, 1998.

[60] A. G. Lynch, D. E. Neal, J. D. Kelly, G. J Burtt, and N. P. Thorne. Missing channels in two-colour microarray experiments: combining single-channel and two-channel data. *BMC Bioinformatic*, 8:26, 2007.

[61] E. Marshall. Getting the noise out of gene arrays. *Science*, 306:630–631, 2004.

[62] D. P. Martin, C. Williamson, and D. Posada. Rdp2: recombination detection and analysis from sequence alignments. *Bioinformatics*, 21:260–262, 2005.

[63] I. Milne, F. Wright, G. Rowe, D. F. Marshall, D. Husmeier, and G. McGuire. Topali: software for automatic identification of recombinant sequences within DNA multiple alignments. *Bioinformatics*, 20:1806–1807, 2004.

[64] A. Milosavljevia. Discovering sequence similarity by the algorithmic significance. In *In Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 284–291, 1993.

[65] R. E. Myers, C. V. Gale, A. Harrison, Y. Takeuchi, and P. Kellam. A statistical model for HIV-1 sequence classification using the subtype analyser (STAR). *Bioinformatics*, 21:3535–3540, 2005.

[66] C. L. Nutt, D. R. Mani, R. A. Betensky, P. Tamayo, J. G. Cairncross, C. Ladd, U. Pohl, C. Hartmann, M. E. McLaughlin, T. T. Batchelor, P. M. Black, A. V. Deimling, S. L. Pomeroy, T. R. Golub, and D. N. Louis. Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Research*, 63:1602–1607, 2003.

[67] S. Oba, M. Sato, I.Takemasa, M. Monden, K. Matsubara, and S. Ishii. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19:2088–2096, 2003.

[68] M. Ouyang, W. J. Welsh, and P. Georgopoulos. Gaussian mixture clustering and imputation of microarray data. *Bioinformatics*, 20:917–923, 2004.

[69] J. Qi, B. Wang, and B. Hao. Whole proteome prokaryote phylogeny without sequence alignment: a k-string composition approach. *Journal of Molecular Evolution*, 58:1–11, 2004.

[70] A. Rambaut, D. Posada, K. A. Crandall, and E. C. Holmes. The causes and consequences of HIV evolution. *Nature Reviews Genetics*, 52:52–61, 2004.

[71] E. Rivals, M. Dauchet, J. Delahaye, and O. Delgrange. Compression and genetic sequences analysis. *Biochimie*, 78:315–322, 1996.

[72] N. Saitou and M. Nei. The Neighbor-Joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[73] K. Saunders, A. M. King, D. McCahon, J. W. I. Newman, W. R. Slade, and S. Forss. Recombination and oligonucleotide analysis of guanidine-resistant foot-and-mouth disease virus mutants. *Journal of Virology*, 56:921–929, 1985.

[74] I. Scheel, M. Aldrin, I. K. Glad, R. Sorum, H. Lyng, and A. Frigessi. The influence of missing value imputation on detection of differentially expressed genes from microarray data. *Bioinformatics*, 21:4272–4279, 2005.

[75] D. R. Schwartz, S. L. R. Kardia, K. A. Shedden, R. Kuick, G. Michailidis, J. M. G. Taylor, D. E. Misek, R. Wu, Y. Zhai, D. M. Darrah, H. Reed, L. H. Ellenson, T. J. Giordano, E. R. Fearon, S. M. Hanash, and K. R. Cho. Gene expression in ovarian cancer reflects both morphology and biological behavior, distinguishing clear cell from other poor-prognosis ovarian carcinomas. *Cancer Research*, 62:4722–4729, 2002.

[76] M. S. B. Sehgal, L. Gondal, and L. S. Dooley. Collateral missing value imputation: a new robust missing value estimation algorithm for microarray data. *Bioinformatics*, 21:2417–2423, 2005.

[77] H. Shatkay, S. Edwards, W. J. Wilbur, and M. Boguski. Genes, themes and microarray: using information retrieval for largescale gene analysis. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 317–328, 2000.

[78] S. K. Shevade and S. S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19:2246–2253, 2003.

[79] Y. Shi, Z. Cai, L. Xu, W. Ren, R. Goebel, and G. Lin. A model-free greedy gene selection for microarray sample class prediction. In *Proceedings of 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2006)*, pages 406–413, 2006.

[80] M. A. Shipp, K. N. Ross, P. Tamayo, A.P Weng, J. L. Kutok, R. C. T. Aguiar, M. Gaasenbeek, M. Amgel, M. Reich, G. S. Pinkus, T. S. Ray, M. A. Kovall, K. W. Last, A. Norton, T. A. Lister, J. Mesirov D. S. Neuberg, E. S. Lander, J. C. Aster, and T. R. Golub. Diffuse large b-cell lymphoma outcome prediction by gene expression profiling and supervised machine learning. *Nature Medicine*, 8:68–74, 2002.

[81] D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D'Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:203–209, 2002.

[82] B. Snel, P. Bork, and M. Huynen. Genome evolution: gene fusion versus gene fission. *Trends in Genetics*, 16:9–11, 2000.

[83] B. Snel, P. Bork, and M. A. Huynen. Genome phylogeny based on gene content. *Nature Genetics*, 21:108–110, 1999.

[84] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[85] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21:631–643, 2005.

[86] G. W. Stuart and M. W. Berry. A comprehensive whole genome bacterial phylogeny using correlated peptide motifs defined in a high dimensional vector space. *Journal of Bioinformatics and Computational Biology*, 1:475–493, 2003.

[87] G. W. Stuart and K. Moffett. Integrated gene and species phylogenies from unaligned whole genome sequence. *Bioinformatics*, 18:100–108, 2002.

[88] G. W. Stuart, K. Moffett, and R. F. Bozarth. A whole genome perspective on the phylogeny of the plant virus family tombusviridae. *Archives of Virology*, 149:1595–1610, 2004.

[89] G. W. Stuart, K. Moffett, and J. J. Leader. A comprehensive vertebrate phylogeny using vector representation of protein sequences from whole genomes. *Molecular Biology and Evolution*, 19:554–562, 2002.

[90] A. I. Su, J. B. Welsh, L. M. Sapinoso, S. G. Kern, P. Dimitrov, H. Lapp, P. G. Schultz, S. M. Powell, C. A. Moskaluk, H. F. Frierson, Jr., and G. M. Hampton. Molecular classification of human carcinomas by use of gene expression signatures. *Cancer Research*, 61:7388–7393, 2001.

[91] M. Suárez-Farñ, M. Pellegrino, K. M. Wittkowski, and M. O. Magnasco. Harshlight: a "corrective make-up" program for microarray chips. *BMC Bioinformatic*, 6:294, 2005.

[92] M. Suárez-Fariñas, A. Haider, and K. M. Wittkowski. "Harshlighting" small blemishes on microarrays. *BMC Bioinformatics*, 6:65, 2005.

[93] V. V. S. Suryanarayana, B. Madanamohan, P. Bist, C. Natarajan, and J. D. Tratschin. Serotyping of foot-and-mouth disease virus by antigen capture reverse transcriptase/polymerase chain reaction. *Journal of Virological Methods*, 80:45–52, 1999.

[94] I. Takemasa, H. Higuchi, H. Yamamoto, M. Sekimoto, N. Tomita, S. Nakamori, R. Matoba, M. Monden, and K. Matsubara. Construction of preferential cDNA microarray specialized for human colorectal carcinoma: molecular sketch of colorectal cancer. *Biochemical and Biophysical Research Communications*, 285:1244–1249, 2001.

[95] P. H. Tran, D. A. Peiffer, Y. Shin, L. M. Meek, J. P. Brody, and K. W. Cho. Microarray optimizations: Increasing spot accuracy and automated identification of true microarray signals. *Nucleic Acids Research*, 30:e54, 2002.

[96] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17:520–525, 2001.

[97] O. G. Troyanskaya, M. E. Garber, P. O. Brown, D. Botstein, and R. B. Altman. Nonparametric methods for identifying differentially expressed genes in microarray data. *Bioinformatics*, 18:1454–1461, 2002.

[98] H. van Rensburg, D. Haydon, F. Joubert, A. Bastos, L. Heath, and L. Nel. Genetic heterogeneity in the foot-and-mouth disease virus leader and 3c proteinases. *Gene*, 289:19–29, 2002.

[99] X. Wu, Z. Cai, X.-F. Wan, T. Hoang, R. Goebel, and G. Lin. Nucleotide composition string selection in HIV-1 subtyping using whole genomes. *Bioinformatics*, 23(14):1744–1752, 2007.

[100] X. Wu, X. Wan, G. Wu, D. Xu, and G. Lin. Whole genome phylogeny construction via complete composition vectors. *International Journal on Bioinformatics Research and Applications*, 2:219–248, 2006.

[101] M. Xiong, X. Fang, and J. Zhao. Biomarker identification by feature wrappers. *Genome Research*, 11:1878–1887, 2001.

[102] K. Yang, Z. Cai, J. Li, and G. Lin. A stable model-free gene selection in microarray data analysis. *BMC Bioinformatics*, 7:228, 2006.

[103] X. Zhou, K. Y. Liu, and S. T. C. Wong. Cancer classification and prediction using logistic regression with Bayesian gene selection. *Journal of Biomedical Informatics*, 37:249–259, 2004.