

**INTEGRATED FEATURE SUBSET
SELECTION/EXTRACTION WITH
APPLICATIONS IN BIOINFORMATICS**

By

Xian Xu

August 16, 2006

A DISSERTATION SUBMITTED TO THE
FACULTY OF THE GRADUATE SCHOOL OF STATE
UNIVERSITY OF NEW YORK AT BUFFALO
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

© Copyright 2006

by

Xian Xu

To my dear wife, without her infinite patience and love, the completion of this thesis is simply impossible. And to my parents, for obvious reasons.

Acknowledgement

Doing research is not always fun. Sometime I'd rather be playing video games. My advisor Professor Aidong Zhang provided me constant support, both in academic training and in four years of financial support. From her I learnt the meaning of dedication and professionalism. I can not thank her more.

I also want to thank the Department of Computer Science and Engineering in the State University of New York at Buffalo, for the financial support during my study. Many thanks to Professor Murali Ramanathan and Professor Matthew Beal for sitting in my Ph.D. committee and gave me many valuable suggestions and questions. Thanks to Professor Yulan Liang for being outside reader for my thesis. Thanks to my colleagues Dr. Liangjiang Wang, Dr. Pengjun Pei, Dr. Daxin Jiang and Dr. Wei Wang for discussion of ideas and support.

Abstract

Feature subset selection and extraction algorithms are actively and extensively studied in machine learning literature to reduce the dimensionality of feature space, since high dimensional data sets are generally not efficiently and effectively handled by a large array of machine learning and pattern recognition algorithms. When we stride into the analysis of large scale bioinformatics data sets, such as microarray gene expression data sets, the high dimensionality of feature space compounded with the low dimensionality of sample space, creates even more problems for data analysis algorithms.

Two foremost characteristics of microarray gene expression data sets are: 1. the correlation between features (genes) and 2. the availability of domain knowledge in computable format. In this dissertation, we will study effective feature selection and extraction algorithms with applications to the analysis of the new emerging data sets in the bioinformatics domain. Microarray gene expression data set, the result of large scale RNA profiling techniques, is our primary focus in this thesis. Several novel feature (gene) selection and

extraction algorithms are proposed to deal with peculiarities on microarray gene expression data set.

To address the first characteristic of the microarray gene expression data set, we first propose a general feature selection algorithm called Boost Feature Subset Selection (BFSS) based on permutation analysis to broaden the scope of selected gene set and thus improve classification performance. In BFSS, subsequent features to be selected focus on those samples where previously selected features fail. Our experiments showed the benefit of BFSS for t-score and S2N (signal to noise) based single gene scores on a variety of publicly available microarray gene expression data sets.

We then examine the correlations among features (genes) explicitly to see if such correlations are informative for the purpose of sample classification. This results in our gene extraction algorithm called virtual gene. A virtual gene is a group of genes whose expression levels are combined linearly. The combined expression levels of a virtual gene instead of the real gene expression levels are used for sample classification. Our experiments confirm that by taking into consideration the correlations between gene pairs, we could indeed build a better sample classifier.

Microarray gene expression data set only represents one aspect of our knowledge of the underlying biological system. Currently there are lots of biological knowledge in computable format that can be accessed from Internet. Continue to address the second characteristic of

the microarray gene expression data set, we investigate the integration of domain knowledge, such as those imbedded in gene ontology annotations, for the use of gene selection and extraction. GO annotation enables us to investigate correlations among bigger groups of genes in an informed way and thus expand beyond pairwise virtual gene algorithm. Correlations between biologically related genes are examined and used for sample classification. Our experiments showed considerable improvement in the term of classification accuracy. GO annotations also enable us to suppress false positives in selected gene set, which becomes an increasing problem for gene selection algorithms on microarray gene data set due to the limited number of samples.

List of Figures

1.1.1 The k-nearest neighbor algorithm. A set of positive and negative labeled data points are shown in figure a). The query point is x_q . Using a 1-nearest neighbor algorithm, x_q will be classified as a negative instance since its nearest neighbor is a negative instance. However, when considering three nearest neighbors of x_q , two of them are positive labeled. Thus using 3-nearest neighbor algorithm, x_q should be labeled as positive. Figure b) in the right shows the potential decision boundary for 1-nearest neighbor algorithm of five data points. Each line is a potential decision boundary depending on how the data points are labeled. 11

1.1.2 Algorithm: K-nearest neighbor algorithm. 12

1.1.3 FLD classifier. The bigger figure in left bottom shows original data plotted on a 2D surface. The bold line represents the FLD projection direction and thinner line represents the decision boundary formed by an FLD classifier. The smaller figure on top right corner shows the 1D layout of data points after data being projected onto the FLD projection direction.	13
1.1.4 The basic idea of an SVM's maximal margin classifier. Decision boundary is a hyperplane in feature space that separates sample of different class labels by maximal margin. The figure illustrates a linearly separable 2D case. When data points are not linearly separable, an SVM classifier uses the kernel trick to project data points onto a higher dimension feature space, in which projected data points become linearly separable.	19
1.1.5 Using kernel function to map problems that are not linearly separable in the input space to higher dimension feature space where patterns become linearly separable.	20
1.3.1 DNA is formed by coupling the nucleotides between the phosphate group from a nucleotide (which is positioned on the 5th C-atom of the sugar molecule) with the hydroxyl on the 3rd C-atom on the sugar molecule of the previous nucleotide [122]. A-G-C-T form the four bases of a DNA molecular.	34

1.3.2 The table of all twenty amino acids found in proteins [3]. Their names and chemical makeups.	35
1.3.3 The central dogma of molecular biology. This figure illustrates the process of DNA replication, DNA transcription into messenger RNA and messenger RNA translation into protein [122].	37
1.3.4 Genetic Coding: Encoding of amino acids using four bases (A,C,U,G) found in RNA molecular. Every three bases form a codon, which encodes one amino acid [4]. Amino acids are the building stones of proteins. The DNA T base is substituted by a RNA U base.	39
1.3.5 The workflow of cDNA microarray experiment. Tissue samples from different sample classes are processed by RT/PCR for mRNA amplification and labeled using fluorescent material. They are then exposed to cDNA chip for hybridization. Resulting chip is then scanned and processed to produce a two dimensional numerical array of microarray gene expression data set that is used by data analysis algorithms. [65]. There are other techniques to build microarrays.	41

1.3.6 The microarray data set after processing. It is a two dimensional numerical array with genes as rows and tissue samples as columns. Samples are labeled by some external labels, such as normal or cancer.	42
1.3.7 The GO hierarchy. The figure shows all GO terms leading to GO term GO:0003700, or “transcription factor activity”. This GO term lies in the molecular function branch of gene ontology. GO terms form a DAG (directed acyclic graph). There are two paths leading from root GO node to GO:0003700, either from the GO term “binding” or through the GO term “transcription regulator activity” [96].	43
1.3.8 The GO annotations are provided by participating databases [27], such as Flybase [45] and SGD [105]. This figure illustrates the annotations of various genes and gene products using one GO term GO:0005388, or “calcium transporting ATPase activity”, by different biological database groups. . . .	45

2.1.1 An illustrating example for the BFSS algorithm: redundancy in a selected gene set. Genes with high individual discriminative scores may not be overall good choices for a gene subset. Gene 1 and Gene 2 have higher individual discriminative scores, yet their expression levels are highly correlated across different experimental samples. Gene 3 on the other hand provides new information, although its discriminative score is lower. . . .	55
2.3.1 DLD prediction accuracy on colon cancer data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.	68
2.3.2 KNN (k=3) prediction accuracy on colon cancer data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.	69
2.3.3 SVM prediction accuracy on colon cancer data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.	70
2.3.4 DLD prediction accuracy on leukemia data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.	72

2.3.5 KNN (k=3) prediction accuracy on leukemia data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.	73
2.3.6 SVM prediction accuracy on leukemia data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.	74
2.3.7 DLD prediction accuracy on multi-class data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.	77
2.3.8 KNN (k=3) prediction accuracy on multi-class data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.	78
2.3.9 SVM prediction accuracy on multi-class data set for four different feature selection algorithms: t-score, S2N, boost t-score and boost S2N.	79

3.1.1 The idea behind the virtual gene algorithm: examples of gene pair being better predictor of class labels than any constituent single gene. Both a synthetic and a real world example are given here. The real world example comes from colon cancer data set [8]. In the figures, two genes are not good predictor of sample class labels individually. However, when combined, they become much better predictors.	85
3.4.1 Result of experiment alon.1. Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on colon cancer data set using DLD classifier. Left figure shows prediction accuracy against the number of genes used to build DLD classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.	94
3.4.2 Result of experiment alon.2. Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on colon cancer data set using KNN classifier (k=3). Left figure shows prediction accuracy against the number of genes used to build KNN classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.	95

3.4.3 Result of experiment along.3. Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on colon cancer data set using SVM classifier. In this experiment, we used a radial kernel for SVM. Left figure shows prediction accuracy against the number of genes used to build SVM classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.	96
3.4.4 Prediction accuracy and its standard deviation of KNN (k=3) using different number of clusters in k-means algorithm (stage 1 of algorithm 4) on colon cancer data set. The prediction accuracy degrades as the number of clusters increase. However, the within-cluster gene pairs (256-cluster version vs. 8-cluster version) retain much information as a reduction of 99.9% of pairs results only around 2% decrease in prediction accuracy.	100
3.4.5 The boxplot of mean KNN (k=3) classification accuracy using <i>pairwise virtual gene</i> algorithm with 20 different initial clusters on colon cancer data set.	101

3.4.6 Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on leukemia data set using DLD classifier. Left figure shows prediction accuracy against the number of genes used to build DLD classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes. 104

3.4.7 Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on leukemia data set using KNN classifier (k=3). Left figure shows prediction accuracy against the number of genes used to build KNN classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes. 105

3.4.8 Prediction accuracy of 4 feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on multi-class data set using KNN classifier (k=3). Left figure shows prediction accuracy against the number of genes used to build KNN classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes. 108

3.4.9 Prediction accuracy of 4 feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on multi-class data set using DLD classifier. Left figure shows prediction accuracy against the number of genes used to build KNN classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.	109
4.2.1 Main data structures used in Algorithms 5 (<i>GO_gene_counter</i>) and Algorithm 6 (Best GO Adjusted Score).	121
4.2.2 Setup of experiment data set for studying the alleviation of false positive rate on microarray expression data set.	125
5.1.1 Structure of gene ontology and its annotation. Solid lines between genes and GO terms indicate direct annotation and dotted lines indicate inferred annotation by the property of transitivity of GO annotations.	134
5.2.1 Flowchart of our virtual gene feature extraction algorithm for sample classification on microarray data set by integrating domain knowledge in the form of GO annotations.	139

List of Tables

- 2.3.1 Classification accuracy (%) without FSS on three publicly available data sets: colon cancer, leukemia and multi-class. 67

- 4.1.1 Gene selection on randomized data set. Each entry shows the number of genes that score greater than cutoff t-scores in each data set (original and random generated.) 114

- 4.3.1 Performance of GO adjusted scores as measured in false positive rate using S2N score. 128

- 4.3.2 Performance of GO adjusted scores as measured in false positive rate using t-score. 129

5.3.1 Performance (classification accuracy) of GO based virtual gene feature extraction algorithm (GOF), compared with single gene based algorithms: t-test score (tscore) [15], signal-to-noise ratio (S2N) [53], on colon cancer data set. Numbers in parenthesis are the number of genes. <i>vg</i> stands for virtual gene and <i>rg</i> stands for real gene. We also report classification performance when no feature selection algorithm is used (ALL column).	147
5.3.2 Standard deviation of classification accuracy of GO based virtual gene feature extraction algorithm (GOF), compared with single gene based algorithms: t-test score (tscore) [15], signal-to-noise ratio (S2N) [53], on colon cancer data set.	147
5.3.3 Performance (classification accuracy) of GO based virtual gene feature extraction algorithm (GOF), compared with single gene based algorithms: t-test score (tscore) [15], signal-to-noise ratio (S2N) [53], on leukemia data set. Numbers in parenthesis are the number of genes. <i>vg</i> stands for virtual gene and <i>rg</i> for real gene. We also report classification performance when no feature selection algorithm is used (ALL column).	150

5.3.4 Standard deviation of classification accuracy of GO based virtual gene feature extraction algorithm (GOF), compared with single gene based algorithms: t-test score (tscore) [15], signal-to-noise ratio (S2N) [53], on leukemia data set. 150

List of Algorithms

1	<i>WorstSampleSet</i> : Calculate the worst set of samples using a greedy algorithm	59
2	<i>BFSS</i> : Boost Feature Subset Selection	60
3	<i>gen_vg</i> : Calculating Virtual Gene From Training Data	86
4	<i>pairwise_vg</i> : Pairwise Virtual Gene Selection	88
5	<i>GO_gene_counter</i> ($go_{root}, scores, \theta, \epsilon, \leftarrow$) Compute genes/informative genes for each GO term	122
6	<i>bS</i> ($go, scores, G, \beta, \gamma$): Best GO Adjusted Score	123

Contents

Acknowledgement	i
Abstract	ii
List of Figures	iv
List of Tables	xiv
List of Algorithms	xvii
1 Introduction	1
1.1 A Gentle Introduction to Data Mining and Machine Learning	4
1.1.1 Machine Learning and Data Mining	4

1.1.2	Feature Space	6
1.1.3	Training and Testing Data Sets	8
1.1.4	Classifiers	9
1.1.5	Estimate Classification Accuracy	18
1.2	Feature Selection and Extraction Algorithms	24
1.2.1	What Does Feature Selection Do?	25
1.2.2	Feature Subset Selection Based on Single Feature Discriminative Scores	26
1.2.3	Correlation Based Feature Selection	28
1.2.4	General Feature Subset Selection Algorithms	30
1.2.5	Feature Extraction Algorithms	31
1.3	Some Biology Background	32
1.3.1	DNA, Genes and Proteins	33
1.3.2	Microarray Experiment	38

1.3.3	Gene Ontology and Gene Annotations	42
1.4	Gene Selection for Microarray Experiments	46
1.4.1	Formulation of Our Problem	46
1.4.2	The Limitations of Existing Gene Selection Algorithms and Our Proposed Approaches	49
1.5	Organization of This Thesis	50
2	BFSS: Boost Feature Subset Selection	52
2.1	A Motivating Example	54
2.2	BFSS: Boost Feature Subset Selection	57
2.2.1	BFSS: Boost Feature Subset Selection Algorithm	57
2.2.2	Computational Complexity of BFSS	62
2.3	Experiments	63
2.3.1	Data Sets and Data Preparation	64
2.3.2	Feature Selection and Classification Algorithms	65

2.3.3	Experiments on Colon Cancer Data Set	67
2.3.4	Experiments on Leukemia Data Set	72
2.3.5	Experiments on Multi-class Cancer Data Set	75
2.4	Conclusion and Discussion	76
3	Virtual Gene: Correlation Base Gene Selection	82
3.1	An Example: The Problem of Single Feature Based Discriminative Score	83
3.2	Virtual Gene and Pairwise Virtual Gene Algorithm	84
3.3	Complexity of the Pairwise Virtual Gene Algorithm	89
3.4	Experiments	90
3.4.1	Colon Cancer Data set	91
3.4.2	Leukemia Data Set	102
3.4.3	Multi-class Cancer Data Set	106
3.5	Conclusion and Future Work	107

4	Dealing with False Positives Using Gene Ontology	112
4.1	The Problem of Gene Selection on Small Sized Samples	113
4.2	Integrating Biological Knowledge into Gene Selection	115
4.2.1	GO Adjusted Scores	116
4.2.2	Complexity of Best GO Adjusted Score	120
4.2.3	Experiment Setup	124
4.3	Experiments	126
4.4	Conclusion and Future Work	128
5	Integration of Gene Ontology with Virtual Gene	132
5.1	The Hierarchical Structure in Feature Space	133
5.2	Feature Extraction by Integrating Ontology Distance with Gene Expression	
	Data Set	138
5.3	Experiments and Discussion	144
5.4	Conclusion	149

6 Conclusion of the Thesis	153
7 List of Publications	162

Chapter 1

Introduction

It is widely believed that thousands of genes and their products in a living organism function in a complicated and orchestrated way that creates the mystery of life [2]. Molecular biology used to be about one gene at a time. However, with the completion of genome sequencing project for several species [44, 37, 60] and the advent of microarray technology for large scale RNA profiling [128], it becomes possible to quantize molecular states of a living cell. Although crude, such technique offers biologist first time in history a look at biological system at genome level. Biology is transforming from a data poor discipline into a data rich one. However, efficient analysis of the large amount of data generated by various biological protocols has proven to be even more difficult than collecting such data itself.

Bioinformatics as an inter-discipline field, is gaining traction. Broadly speaking, bioinformatics describes any use of computers to handle biological information. In a narrower sense, bioinformatics is more related to computational molecular biology: use computer programs to characterize molecular components of living things [1].

Among the various mountains of data collected during the course of last decade, microarray gene expression data set is one of the most common functional data set. The expression levels of thousands of mRNAs (messenger RNA) or ESTs (Expressed Sequence Tag) [78] are measured simultaneously in one experiment. A typical microarray gene expression data set is a two dimensional array of numbers with thousands of rows (genes) and tens to hundreds of columns (experiments). Computational algorithms are needed to extract meaningful information out of the vast amount of data. Microarray data sets are used for studying cell cycle [112], stress response [51, 50], clustering [70], sample classification [111, 8, 22, 73, 131] and phenotype detection [114], etc.. The problem interests me most is the sample classification problem using microarray data set, which is the main focus of this dissertation. The expression levels of thousands of genes span a feature space for each sample tissue, characterizing its molecular state. Sample classification algorithms using microarray data set try to automatically discover a model to explain the correlation between the molecular states of the tissues and the phenotypes of the tissues. If such a model can be successfully built, it can be used to diagnose disease more accurately and to discover new subtypes of a disease, for example.

Emerging data sets such as the microarray gene expression data sets pose special challenge for pattern recognition algorithms. The main obstacle is the limited number of samples due to practical and financial concerns. This results in the situation where the number of features (or genes) well outnumbers the number of observations. The term “curse of dimensionality” and “peaking phenomenon” are coined in the machine learning and pattern recognition community, referring to the phenomenon that inclusion of excessive features may actually degrade the performance of a classifier if the number of training examples used to build the classifier is relatively small compared to the number of features [69]. Typical treatment is to reduce the dimensionality of feature space before classification using feature selection and feature extraction algorithms. Feature extraction algorithms create new features based on transformation and/or combination of original features while feature selection algorithms aim to select a subset of original features. Those new/selected features preserve most variation of data in the feature space. Techniques like PCA (principal component analysis) and SVD (singular value decomposition) [10] have been used to create salient features [59, 73] for sample classification on microarray data sets. Feature selection, or in our case, gene selection generates a small set of informative genes, which not only leads to better classifiers, but also enables further biological investigation.

In this chapter, both biological and computational background surrounding our work will be presented to familiarize readers on these subjects. We will introduce the concept of feature space, classifiers, feature selection and extraction algorithms. We will also discuss how to

estimate the performance of classifiers. On the biological front, basic ideas of DNA, RNA, protein and microarray experiment will be explained. The central dogma of molecular biology will be also presented. A formal formulation of the problem of feature (gene) selection on microarray gene expression data set is presented at the end of this chapter. And finally, we will discuss the pitfalls of current gene selection algorithms used for the microarray gene expression data analysis and briefly discuss our contributions in this area.

1.1 A Gentle Introduction to Data Mining and Machine Learning

1.1.1 Machine Learning and Data Mining

To learn is to improve automatically with experience [87]. Machine learning, as a broad subfield in artificial intelligence, is concerned with algorithms that could learn from experience. Among the vastly broad fields in machine learning research, classification, or pattern recognition, is one of the most important class of problem studied. A pattern recognition algorithm is concerned with finding models which could be used to automatically classify new instances of unseen data. It is sometimes called supervised learning, as compared to

other learning paradigms where the eventual task of learning is not to assign some predefined labels to data. Different classification algorithms have been widely studied, such as decision trees, artificial neural networks, Bayesian learning, k-nearest neighbor learning, support vector machines, etc. [87]. I will not be able to give detailed account for each of those classifiers. I will briefly discuss some of these learning algorithms that are used in this thesis later in this chapter.

On the contrary to supervised learning, clustering is often called unsupervised learning in the sense that no extra labels are associated with each object. Unsupervised learning aims to find patterns that exist in the data. Clustering is widely used in microarray expression data analysis, to discover broad patterns in such data set. The most prevalent approaches for gene clustering include hierarchical clustering [43], k-means [63], self organizing maps (SOM) [113] and subspace clustering [123]. In those cases, the expression levels of genes across different samples are used as features for genes. By clustering in the gene space, meaningful co-expressed gene groups can be identified [70]. Other approaches to mining gene expression data are also possible. For example, Tang [114] tried to discover hidden phenotype structures underlying gene expression data. In their approach, like in sample classification, each sample tissue is an object and expression levels of different genes are viewed as features. This is one characteristic of microarray expression data set: the data set itself can be viewed either from the gene space, or from the feature space, or from both spaces as in subspace clustering. Still, a more interesting problem is presented in

[71] where an extra dimension is added to microarray gene expression data set: the time dimension. Each cell in this three dimensional data set is the expression level of a certain gene, in a certain tissue sample, at a certain specific time.

Data mining is a new discipline that is under rapid development in recent years. Data mining aims to find novel, previously unknown and useful information from data. It includes for example associate rules mining [6, 137]. A usually given example for associate rule mining is the placement of products in a grocery store in such a way that those products that tend to be purchased together by a customer are placed near each other to boost sales. Association rules mining had been used to discover interesting rules from microarray data set in [26]. Clustering algorithm mentioned in the previous paragraph is also a main focus for data mining researches.

Please refer to Tom Mitchell's *Machine Learning* [87] and Jiawei Han's *Data Mining: Concepts and Techniques* both first and second edition [57, 58] for more detailed treatment on those two subjects.

1.1.2 Feature Space

Machine learning and data mining algorithms deal with every day objects, drawing conclusions (classification and regression) or searching for patterns based on observations. Real

life objects require a consistent way of representation, such that same algorithm could be applied to different data instances. In the literature of machine learning and data mining, real life objects are usually represented using features, such that the properties of real life objects can be manipulated and analyzed. Features can be numerical (continuous or discrete), or categorical. Features, describing the states of objects under study, span a space of all possible states of the object being modeled. Such space is often called the feature space of a learning problem.

Feature space is usually represented as a d -dimensional space, where d is the number of features. Depending on whether class labels are used or not, feature selection algorithms can be categorized as supervised or unsupervised. In the case of supervised feature selection, the goal of feature selection algorithms is to choose a set of features that allows patterns belonging to different classes to occupy in rather compact and distinct region in the feature space. In the case of unsupervised feature selection, newly selected features represent most variations amongst the data. Feature extraction algorithms on the other hand transform original feature space into a secondary space such that patterns become clustered in the new feature space. The effectiveness of feature representation is determined by how well patterns can be separated, which is usually measured by the performance of a classifier. Feature selection and extraction algorithms are the main focuses of this thesis.

1.1.3 Training and Testing Data Sets

In order to learn from experience, a set of data called training data set is presented to a learner. The task of the learner is to build a model explaining the variations in the training data set. Another independent data set called testing data set is normally used to test the accuracy of the model a learner learnt. For bioinformatics and medical data sets such as microarray expression data, the number of experiments are often limited due to financial concerns. There are usually not enough data to serve as independent testing data set. This problem is circumvented by using cross validation algorithms and bootstrap algorithms as shown later in this chapter.

Data sets are represented as data points in the multi-dimensional feature space. In the case of supervised learning (classification), each data point is also associated with a class label. When each of the data point is associated with a numerical label, the learning problem becomes a regression problem. Regression in the statistics community stands for approximation of a real valued function. The construction of efficient and effective learners is an active research area for a long time. In the following sub-sections, we will introduce some of the learners we used in this thesis. The performance of feature selection algorithms is measured by the performance of sample classification later in this thesis.

1.1.4 Classifiers

Depending on the shape of decision boundary learnt, classifiers can be categorized as either linear or as non-linear. It might look like non-linear classifiers are more powerful learners because of their exotic decision boundaries. However, due to limited training data, it is not always a good idea to use powerful classifiers. Intuitively, when a learner tries to fit a model too exactly according to the limited training data set, it generally loses its generalization power on unseen samples. This problem has been extensively studied in statistical learning theories [121]. Statistical learning theory, or VC (Vapnik-Chervonenkis) theory, shows that it is crucial to restrict the class of functions that the learning machine can implement to one with a capacity that is suitable for the amount of available training data [62]. Otherwise, the learnt model is tuned to the training data too much, sacrificing its power to generalize to new unseen data. This problem is called “overfitting” in machine learning literature. Overfitting is generally recognized to be a violation of Ockham’s razor principle, which states that the explanation of any phenomenon should make as few assumptions as possible.

There are generally two stages of a learning algorithm: learning and classification. In the learning stage, a learner is presented with a set of training samples, from which the learner tries to find a conforming model. After a model is successfully learnt, a learner can then be used to classify new unseen and unlabeled instances of data. It will label the unseen data according to the model it learnt in the previous learning stage. Learners differ vastly

in how they operate in those two stages. Some are fast in learning yet slow in classifying; while others are slow in learning yet fast in classifying when a model is already learnt.

In this section, we will introduce three vastly different classifiers: KNN (K-Nearest Neighbor), FLD (Fisher Linear Discriminant) and SVM (Support Vector Machine). The performance of our feature selection and extraction algorithms are later measured using those three classifiers. The very nature of differences in those classifiers ensure that our proposed feature selection and extraction algorithms are not tuned to some specific classification algorithms.

KNN Classifier

The k-nearest neighbor classifier is an instance based classifier [87]. It was originally proposed by Cover, etc. in [115]. Instead of learning an explicit model, instance based classifiers simply store training data and defer the generalization beyond training instances until a new instance must be classified. Instead of trying to find a global model, instance based methods produce classifiers that are local for each new instance. Since the learning is actually just storing training data internally instance based learning algorithms, such as KNN, is very fast in learning.

Assuming objects are characterized by an n-dimensional space \mathcal{R}^n . An object x is described

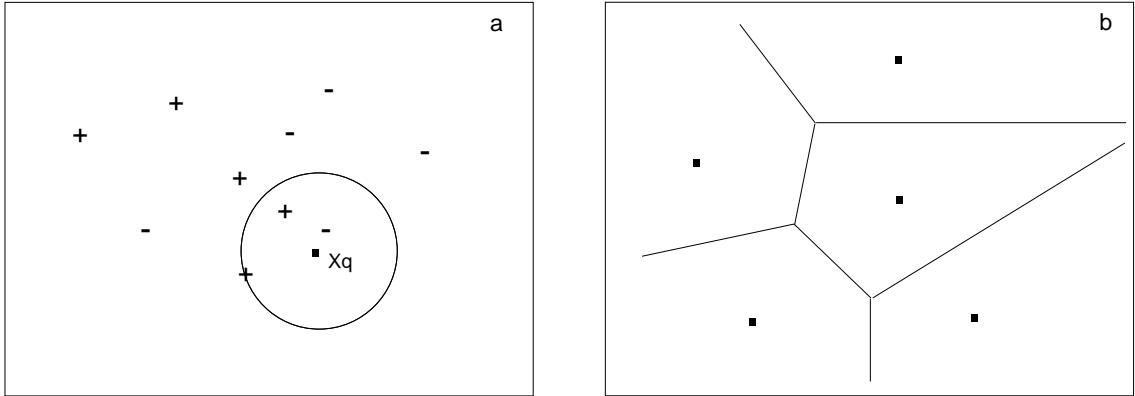


Figure 1.1.1: The k -nearest neighbor algorithm. A set of positive and negative labeled data points are shown in figure a). The query point is x_q . Using a 1-nearest neighbor algorithm, x_q will be classified as a negative instance since its nearest neighbor is a negative instance. However, when considering three nearest neighbors of x_q , two of them are positive labeled. Thus using 3-nearest neighbor algorithm, x_q should be labeled as positive. Figure b) in the right shows the potential decision boundary for 1-nearest neighbor algorithm of five data points. Each line is a potential decision boundary depending on how the data points are labeled.

by an n -dimensional feature vector $(a_1(x), a_2(x), \dots, a_n(x))$. Euclidean distances $d(x_i, x_j)$ between two objects x_i and x_j is defined as

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2} \quad (1.1.1)$$

Each of the data instance x is associated with a label $f(x)$. For a new query data item x_q , the goal of classification is to find the value $f(x_q)$. The KNN classification algorithm is described in Algorithm 1.1.2 and illustrated in Figure 1.1.1. The label of x_q is determined by a majority vote of x_q 's k nearest neighbors in the sense of Euclidean distance (Equation 1.1.1). Other distance functions could also be used in place of the Euclidean distance. The learning stage only involves remembering all training data points. In the classification

stage, first k nearest data points to the query data point x_q are determined. The label that occurs the most often in this neighborhood will be assigned as the label for the new data instance. There will not be ties for two class problem when k is an odd number. Otherwise ties can be broken arbitrarily. KNN classifier is generally fast. Even when training data set is reasonably large. Spatial indexing algorithms, such as R-Tree [54] and R*-Tree [13], can be used to store training data points so that k-nearest neighbor queries in the classification stage can be efficiently answered.

Algorithm KNN training algorithm

For each training example $\langle x, f(x) \rangle$, add it to list *training_examples*.

KNN classification algorithm

Given a query instance x_q to be classified

Let $x_1 \cdots x_k$ denote the k instances from *training_examples* that are nearest to x_q

Return

$$f(x_q) \leftarrow \operatorname{argmax}_{v \in V} (\sum_{i=1}^k \delta(v, f(x_i)))$$

where V is the collection of labels and $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

Figure 1.1.2: Algorithm: K-nearest neighbor algorithm.

The decision boundary of a KNN classifier is a collection of hyperplanes as illustrated in Figure 1.1.1 in the case for one nearest neighbor.

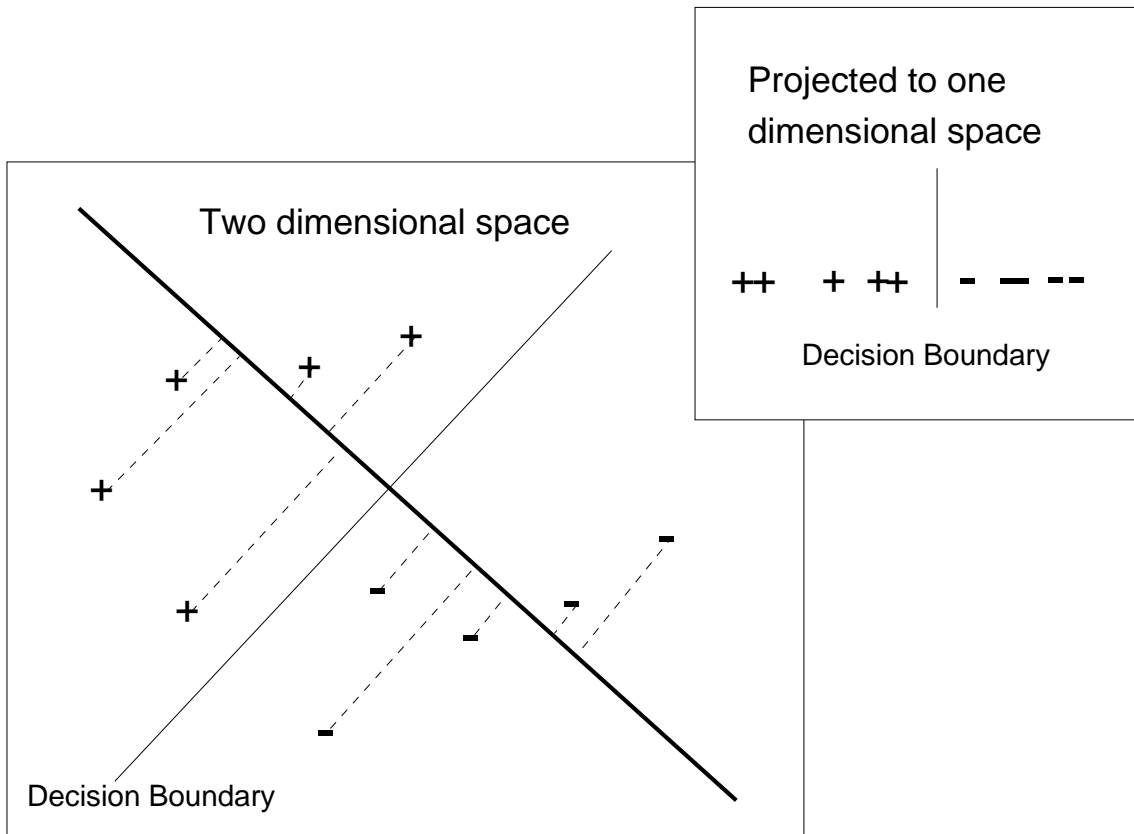


Figure 1.1.3: FLD classifier. The bigger figure in left bottom shows original data plotted on a 2D surface. The bold line represents the FLD projection direction and thinner line represents the decision boundary formed by an FLD classifier. The smaller figure on top right corner shows the 1D layout of data points after data being projected onto the FLD projection direction.

Fisher Linear Discriminant (FLD)

Fisher Linear Discriminant (FLD) can be used as a supervised classification algorithm [35]. FLD aims to find the best linear direction to project data points in an n -dimension space such that the data of different class labels are separated to the maximum. Let us continue to use notations from last section. Let us further define a counter for each class label $C(v)$ to be:

$$C(v) = \sum_{i=1}^k \delta(v, f(x_i)) \quad (1.1.2)$$

Class mean $\mu(v)$ of a label is defined to be the mean vector of observations belonging to label v . We use $x \in v$ to denote the fact that data instance x is labeled by class label v . We also use $a(x)$ to denote the n -dimensional feature vector for object x .

$$\mu(v) = \frac{1}{C(v)} \sum_{\forall x \in v} a(x) \quad (1.1.3)$$

Grand mean μ is the mean vector for all data in the training set. n is the number of data instances in the training set.

$$\mu = \frac{1}{m} \sum_x a(x) \quad (1.1.4)$$

We assume every data point is labeled by one and only one class label. Given this condition, we have:

$$m = \sum_v C(v) \quad (1.1.5)$$

FLD aims to find the linear direction that separates objects from different classes maximally while making objects within each class as compact as possible. This can be formulated to simultaneously minimize the within class scatter while maximizing the between class scatter. The within class scatter can be defined as follows:

$$S_w = \sum_x (a(x) - \mu(f(x)))(a(x) - \mu(f(x)))^t \quad (1.1.6)$$

The between class scatter can be defined as:

$$S_b = \sum_{v \in V} C(v)(\mu(v) - \mu)(\mu(v) - \mu)^t, \text{ where } V \text{ is the collection of labels.} \quad (1.1.7)$$

The criteria to optimize is with respect to a vector V :

$$J(V) = \frac{\det(V^t S_b V)}{\det(V^t S_w V)} \quad (1.1.8)$$

The above equation can be solved by solving the generalized eigenvalue problem [5] $S_b V = \lambda S_w V$. As a special case, when S_w has full rank, the generalized eigenvalue problem can be converted into a standard eigenvalue problem $S_w^{-1} S_b V = \lambda V$.

Assume $|V|$ denote the number of different classes. The generalized eigenvalue problem $S_b u = \lambda S_w u$ has at most $|V| - 1$ eigenvalues. Let $u_1, u_2, \dots, u_{|V|-1}$ be the corresponding eigenvectors. The optimal projection matrix u to project original data to a k -dimensional subspace is given by eigenvectors corresponds to the largest k eigenvalues.

For two class case, the optimal projection matrix u is actually a vector, projecting data onto a line (one dimension). The classification is then done using a simple threshold as shown in Figure 1.1.3. The decision boundary of an FLD classifier is always a hyperplane in the n -dimensional feature space. In the learning stage, the FLD projection direction is calculated based on the training data set. Then in the classification stage, new data instance is projected onto the pre-calculated projection direction and the position of the projected data point is used for classification.

Support Vector Machine (SVM)

Support vector machine (SVM) was introduced by Vapnik and his colleagues in a seminal paper in COLT 1992 [18]. Two components characterize the SVM: a maximal margin classifier and a kernel function to map input space non-linearly to higher dimensional feature space [106]. Figure 1.1.4 illustrates the idea of maximal margin classifier. This is different from other classification methods where the decision boundary is determined by optimizing some global criteria function (such as MMSE, or minimum mean squared error for partition based clustering and $J(V)$ for FLD classifier). Maximal margin classifier makes classification decision only based on boundary patterns (or support vectors). Linear decision boundary can be written as $(w \cdot x) + b = 0, w \in \mathcal{R}^N, b \in \mathcal{R}$, corresponding to decision functions $f(x) = \text{sign}((w \cdot x) + b)$. The optimal hyperplane can be uniquely constructed by solving a constrained quadratic optimization problem. Omitting detail calculation, one critical property of the algorithm is that the calculation only involves dot products between patterns. This is why the kernel trick can be introduced.

When training data is not linearly separable, SVM employs a clever kernel trick to map patterns from input space onto a higher dimensional feature space, as illustrated by figure 1.1.5. The mapping is not done explicitly, but introduced using a kernel function. A kernel function is a function k that for all $x, z \in X$ satisfies

$$k(x, z) = \langle \Phi(x), \Phi(z) \rangle, \quad (1.1.9)$$

where Φ is a mapping from X to an feature space F

$$\Phi : x \rightarrow \Phi(x) \in F. \quad (1.1.10)$$

Replacing inner production in the maximal margin classifier with kernel functions, a SVM is able to draw non-linear decision boundaries when input patterns are not linearly separable.

Support vector machines and kernel based methods have shown great potential to be good classifiers. They are both powerful for the non-linear decision boundaries they can express yet mathematically simple. The concept of kernel functions can be also applied to other linear algorithms, e.g. kernel PCA [102], etc..

1.1.5 Estimate Classification Accuracy

After the construction of a learner, empirically evaluating the accuracy of models is a fundamental problem in machine learning. An accurate estimation of model accuracy is important to compare and choose different models for different tasks. Training error refers to the number of mistakes a learner makes on the training data set while testing error refers to the number of mistakes a learner makes on the testing data set. Training error is also called empirical error, which is a too optimistic estimate of classification error. It is always possible to construct a classifier with zero training error. For example, a learner that records

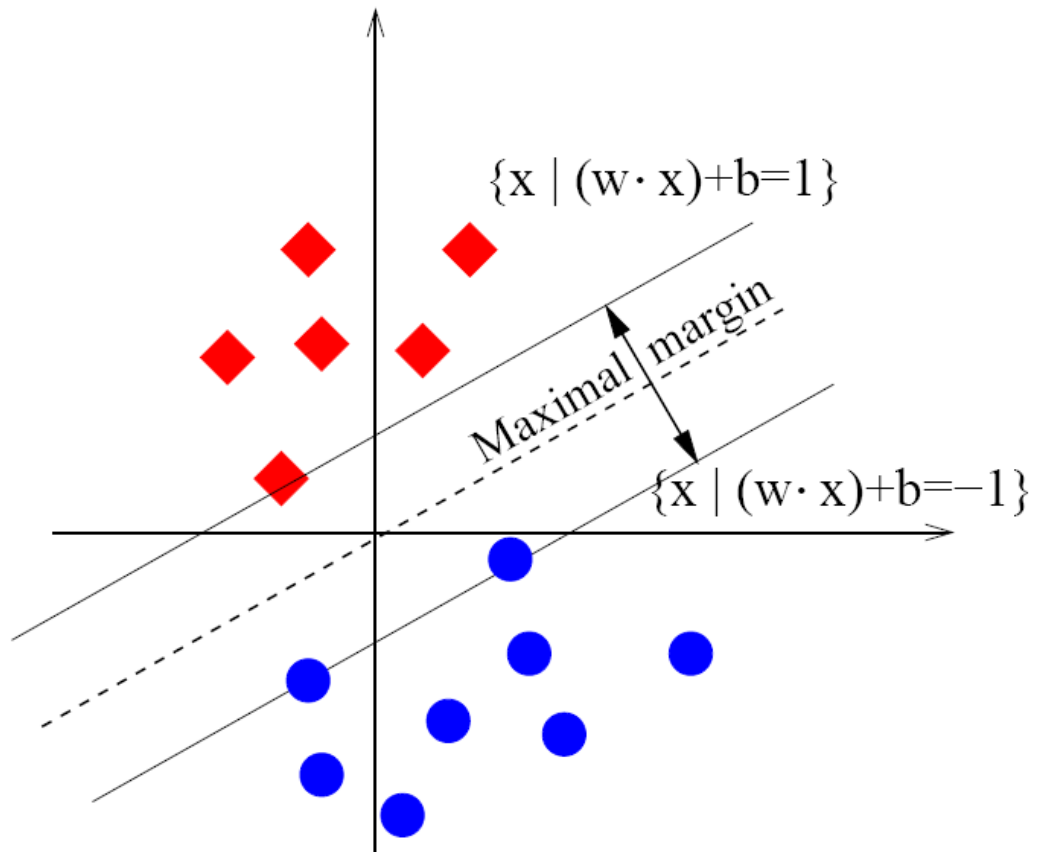


Figure 1.1.4: The basic idea of an SVM's maximal margin classifier. Decision boundary is a hyperplane in feature space that separates sample of different class labels by maximal margin. The figure illustrates a linearly separable 2D case. When data points are not linearly separable, an SVM classifier uses the kernel trick to project data points onto a higher dimension feature space, in which projected data points become linearly separable.

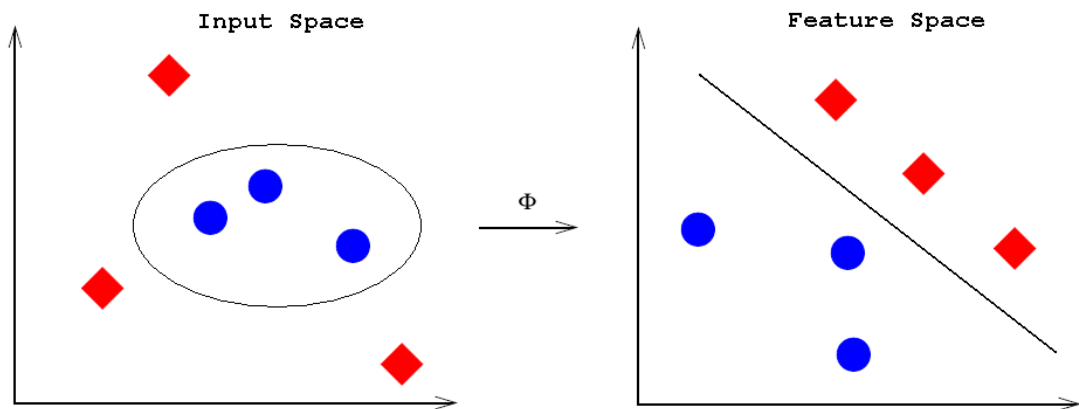


Figure 1.1.5: Using kernel function to map problems that are not linearly separable in the input space to higher dimension feature space where patterns become linearly separable.

all training data set internally and report accordingly will archive zero training error. However, most probably it will not generalize well on the unseen testing data sets. It is also proven theoretically that learners, each performing only slightly better than random, can be combined to form arbitrarily good hypothesis (zero prediction error on the training set) [72, 85]. Thus the construction of a good learner is an art to balance between classification accuracy on training data set and the generalization power of the resultant classifier.

The task for estimation classifier accuracy is further complicated by the fact that the number of samples is often limited. This is especially true in the domain of bioinformatics where preparing extra experiments can be costly. There are two aspects of the problem of limited samples: 1. the limited number of samples limits the statistical significance of the model accuracy observed. 2. the limited number of samples also limit the size of both training and testing data sets. While there is not much “clever” way to circumvent the first problem

besides enlarging the sample size, a lot of research has focused on finding an unbiased estimator of the classification accuracy with low variance even when the size of the samples is small. Since there is only limited samples in the first place for most of microarray expression data sets, separate samples into independent training and testing group in a holdout strategy does not work well.

Cross validation and bootstrapping are two popular estimators for classification accuracy [39, 40, 75, 42, 19, 41]. Molinaro et al. studied classification estimators for small sampled data sets that appear frequently in the bioinformatics domain [88]. Two common cross validation schemes are k-fold cross validation and leave-one-out cross validation. The leave-one-out scheme is a special case of k-fold scheme where k (the number of folds) equals n (the number of samples). In a k-fold cross validation scheme, the available data is partitioned into k mutually exclusive groups. Each group in turn is used as testing data set, while the reminder are used as training data set. The classification accuracy estimates are averaged to provide an overall accuracy estimate. Cross validation produces almost unbiased estimates of classification accuracy, however, the estimate can be highly variable. It can be shown that leave-one-out cross validation bias is in the order of $O(1/N^2)$ where N is the number of samples.

Bootstrap method when used in estimating classification accuracy can be viewed as a smoothed version of cross validation [42]. The simplest bootstrap approach generates B

bootstrap samples $D_1 \cdots D_B$ and estimates model on each of them. Let us assume the models generated are $h_1 \cdots h_B$ accordingly. The classification accuracy is then measured using those models on the original data set. The classification error can be measured using Equation 1.1.11, assuming N to be the number of data points, y_i is the label of data point x_i .

$$E_{bs} = \frac{1}{B} \sum_{b=1}^B \frac{1}{N} \sum_{i=1}^N \delta(y_i - h_b(x_i)) \quad (1.1.11)$$

One problem of this simple bootstrap method is that we might use data points that belongs to training set for testing. In particular it can be shown that the percentage of data points belonging to both training and testing data sets in the simple bootstrap setup to be 63.2%, thus making E_{bs} too optimistic. The remedy is to consider as test cases only those that does not belong to bootstrap sample D_b . Such an estimate is called E_0 and defined in Equation 1.1.12.

$$E_0 = \frac{1}{N} \sum_{i=1}^N \frac{1}{B_i} \sum_{b \in C_i} \delta(y_i - h_b(x_i)), \quad (1.1.12)$$

where C_i is the set of bootstrap samples that do not contain sample i and B_i is the number of such bootstrap samples.

E_0 turns out to be a pessimistic estimate of classification error since the testing cases are all “hard” ones. Intuitively when E_0 is used, the testing set is “hard” because they are totally

new samples that are unknown to the learning process. More reliably and realistically, the .632 estimate combines empirical error with E_0 to produce a less biased bootstrap estimator as defined in Equation 1.1.13.

$$E_{.632} = 0.368 \times E_{emp} + 0.632 \times E_0 \quad (1.1.13)$$

One problem facing both cross validation and bootstrap method is that the distribution of class labels in each random sampling could be significantly different from the original data set. Stratification, or sampling within each class label, is one way to deal with this problem. Stratified versions of cross validation and bootstrap are generally considered to work better.

It is worth noting that when the performance of feature selection algorithms is measured using the classification accuracy of classifiers, the feature selection process must be constrained within the training data set in each turn of cross validation [110]. Otherwise, if feature selection process uses data in the test data set, the result estimate would turn to be too optimistic and is not a good estimator of generation error.

For wrapper based feature subset selection algorithms, where the classification accuracy (or error) of some classifier is as a guidance in feature selection, Sima [109] compared the performance of several error estimators: resubstitution, cross-validation, bootstrap and bolstered error estimation [119] (Braga-Neto reported a comparative study similar to Sima

in [120]). They concluded bolstered works best for feature wrappers from small sized samples, such as microarray data sets.

1.2 Feature Selection and Extraction Algorithms

Traditional pattern recognition algorithms do not work well when the dimensionality of feature space is too high. Feature selection and feature extraction algorithms reduce the dimensionality of feature space while preserving enough information for the underlying learning problem [68]. Genomic studies in bioinformatics often result in high dimensional data set with low sample dimensionality [98]. Feature selection and feature extraction algorithms, especially gene selection and extraction algorithms, are the main focus of this thesis.

Although there are a plenty of more general feature subset selection algorithms proposed in the machine learning literature, gene selection algorithms proposed in the bioinformatics literature are mostly single gene-based in the sense that the criteria used to determine whether a gene is selected or not only deal with expression data of that single gene. Some recent research considers selected gene set as a whole for optimization. One trend is to remove redundant genes from selected gene set. In this section, we will introduce some of the gene selection algorithms proposed in the literature.

1.2.1 What Does Feature Selection Do?

From the machine learning point of view, to overcome the curse-of-dimensionality problem, we can use feature subset selection algorithms to filter out a smaller set of salient genes for classification. From the biologist point of view, smaller set of genes are more easy to explain and conduct further experiments on. Gene selection has several advantages:

- It alleviates the curse of dimension problem and improves classification accuracy when used on microarray data sets [81], [79], [80]).
- It is easier for biologists to gain insight into to smaller set of genes of the underlying genetic mechanism [55].
- It reduces the cost of diagnosis when microarray technique is used in clinics. There is no point of conducting diagnosis on microarray chip with thousands of genes when only a handful of them can successfully diagnose disease [46].

In the following of this section, different feature/gene selection approaches proposed in the literature are surveyed.

1.2.2 Feature Subset Selection Based on Single Feature Discriminative Scores

A lot of researches in the statistics community focused on using some scores measuring discriminant power of single gene in respect to sample class labels [24, 53, 14, 38, 118, 130, 94, 12, 93, 15, 116, 103, 129, 89]. Genes with top scores are selected and combined. We list some of the single gene scores that have been proposed in the literature. The list is by no means complete.

- *t-score*: in two class discriminant analysis, t test is a common statistical test used to test the difference in sample mean. t-score can be used as a measure of how well values are separated in two sets of samples (with different labels) [15]. Assuming two sets of samples coming from same normal distribution with same mean and standard deviation, the larger the t-score is (or the smaller p-value is), the more confident we can conclude that our assumption is actually not correct and the two sets of samples indeed come from two different distribution of different mean (rejecting null hypothesis).
- *S2N*: signal to noise ratio was used in [53] for gene selection. *S2N* is defined as $S2N = \left| \frac{\mu_1 - \mu_2}{\sigma_1 + \sigma_2} \right|$, where μ_1 and μ_2 are the mean value of the respective classes and σ_1 and σ_2 are the standard deviation of the respective classes.

- *TNoM score*: threshold number of misclassifications [14]. Intuition behind this score was that the expression levels of an informative gene should be very different in different sample classes. A simple threshold value (decision stump) would be able separate samples of different classes. $TNoM(g, l) = \min_{(d,t)} Err(d, t|g, l)$ where d, t are the parameters of a decision stump, g is a gene, l is the sample class labels, $Err(d, t|g, l)$ is the number of classification errors made by a decision stump (d, t) .
- *Information Gain*: [129][130] this measure is adapted from information theory directly. Expression levels need to be discretized before applying this score. Information gain is defined as $I(X|Y) = H(X) + H(Y) - H(X, Y)$ where $H(X)$ is the entropy of X . $I(X|Y)$ is biased for excessively multi-valued nominal features. Symmetrical uncertainty $SU(X, Y) = 2(\frac{I(X|Y)}{H(X)+H(Y)})$ can be used for its remedy. Symmetrical uncertainty can be viewed as a normalized version of the information gain measurement.
- *SAM* [118]: significance analysis of microarrays, the uniqueness of SAM is it employs permutation tests to estimate the statistical significance of their single gene-based score.

These discriminative scores are generally easy to compute, with time complexity of $O(n)$.

Note that we do not include computation needed for sorting n genes since it depends on implementation. For example, when top n' out of n genes are selected, we can either sort

n gene scores and pick top n' , which results in $O(n \log n)$ time complexity; or we can find top n' largest gene scores, which can be done $O(n'n)$. We find the total number of discriminative scores an algorithm needs to compute is a more consistent measure of time complexity of gene selection algorithms.

1.2.3 Correlation Based Feature Selection

Feature selection methods based on a single gene ignore correlations between genes. However, it is well known that gene expression levels are correlated with each other. Genes that are regulated by same mechanism exhibit similar expression levels. As a matter of fact, the correlation of expression levels between genes are the basis of gene expression cluster analysis, which is a broad research area by itself [70]. Recent research in feature selection begins to tackle this problem [34, 56, 67, 138, 129, 126, 59]. While trying to include genes with high individual scores, such feature selection algorithms also try to minimize correlations between selected set of genes in the mean time.

- *CBF*: or correlation based feature selection [56]: general search strategy (forward selection, backward elimination, best first) using $M_s = \frac{n\overline{r_{cf}}}{\sqrt{n+n(n-1)\overline{r_{ff}}}}$ as the measure of merit for feature subset. n is the number of features in the feature set. $\overline{r_{cf}}$ is the mean feature-class correlation. $\overline{r_{ff}}$ is the mean feature-feature inter-correlation. The search criterion tries to maximize $\overline{r_{cf}}$ while minimize $\overline{r_{ff}}$.

- *Reduce correlation using clustering* [67]: genes are first clustered using a fuzzy clustering algorithm. Genes having top individual scores in each cluster are selected. The number of genes selected from each cluster depends on the quality of respective gene cluster, which is measured by cluster size and within cluster variation. More genes are selected from big large-variance cluster.
- *Redundancy based feature selection* [138]: Two scoring function were used: individual C -correlation and combined C -correlation. Individual C -correlation is the single gene discriminant score in this thesis. Combined C -correlation is a pairwise score for the correlation of a pair of feature and the class labels. The author used information gain based scores for both individual C -correlation and combined C -correlation. Their RBF algorithm first selects top ranked feature using individual C -correlation. All features whose combined C -correlation score with the selected feature is no better than the selected feature's individual C -correlation are removed from candidate feature set. This process repeats until candidate feature set is empty.
- *EFS*: or efficient feature selection [129]: Feature importance was defined as $Imp(f_m) = B_g(f_m) \times (1 - Corr(f_m, F^*))$, where f_m is an individual feature that has not been selected and F^* is the feature set that has already been selected. B_g is balanced information gain and $Corr$ is a measure of correlation between feature sets. Their EFS algorithm iteratively selects top ranked feature based on its Imp value. Then all Imp values of remaining features in candidate set are recalculated (F^* changes in each

iteration).

- *HykGene* [126]: Hierarchical clustering is performed on top ranked genes. Representative genes from each resulting cluster are then used for further analysis. The assumption made in this work is that genes within the same cluster are correlated and thus removed for redundancy. In this regard, it is similar to the Gene Shaving proposed in [59]

1.2.4 General Feature Subset Selection Algorithms

General feature subset selection algorithms proposed in the machine learning literature have also been used in the analysis of microarray data sets. Although they have been used successfully in other domains, the bioinformatics domain seems slow in picking up these more sophisticated feature subset selection algorithms. Wrappers use classification accuracy on training data sets as a measurement of how well a subset of features performs, thus turning the problem of feature subset selection into an optimization problem. Various search algorithms can be employed to navigate through the exponential space of a feature set's power set. Sequential search algorithm [95] works as a greedy hill-climbing algorithm, including or excluding one feature at a time while maintaining the current best feature subset. It has been used to select informative genes in the context of cancer classification [66]. Relief family (Relief [74], ReliefF [76], RReliefF [99]) [100] is used in microarray data analysis

in [125]).

Siedlecki and Sklansky [107] employed beam search in order to find best feature subset. They [108] also employed genetic algorithm (GA) [86] for the purpose of feature selection. When feature selection criterion function is monotonic, the branch-and-bound (BB) algorithm can be used to find optimal feature subset much more quickly than an exhaustive search [90]. However, in more realistic case, the principle of optimality does not hold for the problem of feature subset selection. Those algorithms tend to be more computational intensive and may not scale well with larger data sets.

Those more general algorithms consider feature correlations to various degree. It addresses the first problem of gene selection we identified in earlier section. However, no algorithm has integrated domain knowledge for the feature selection process.

For a comparison of general purpose feature subset algorithms proposed in machine learning literature, please refer to [68, 33].

1.2.5 Feature Extraction Algorithms

In some application domains, original features may not be effective when used directly. For example, in content based image retrieval systems, it is not meaningful to use each pixel's color value as features. Instead, various new features such as color histogram, various edge

texture features etc. are proposed to describe content of an image. Such feature extraction algorithms are domain specific.

There are also general purpose feature space dimension reduction algorithms that compress original feature space into a lower dimensional space in such a way that most variations in original feature space are preserved. Techniques such as PCA (principal component analysis), SVD (singular value decomposition) [10], SOM (self-organizing map) [61] and FLD (Fisher linear discriminant) are good examples of general purpose feature extraction algorithms.

One drawback of feature extraction algorithms when compared to feature selection algorithms is that the new features created by feature extraction algorithms may not carry any physical meaning of original features. Thus when design new feature extraction algorithms, it makes sense to maintain the meaning of original features in some ways.

1.3 Some Biology Background

It is unavoidable to layout some biological background, as it is the source of peculiarities we are dealing with in this thesis. This section provides a very basic coverage of molecular biology pertaining to the microarray experiment. It is not intended to be comprehensive guide or survey. Please refer to Genes VIII [78] for a complete account on molecular

biology.

1.3.1 DNA, Genes and Proteins

Life on earth began some 3.5 billion years ago, not long after earth itself came into being some 4 billion years ago. All living things on earth share similar structures. The foremost common point is that all hereditary information is passed along in the form of DNA (deoxyribonucleic acid) or RNA (ribonucleic acid) in some cases. A DNA consists of two strands of simpler components, called bases. The bases are so simple that there are actually only four possible choices: adenine (A), guanine (G), cytosine(C), and thymine (T), as shown in Figure 1.3.1. Rapid development in biological technologies over past decades has resulted in complete sequencing of DNA of several model systems, including human's (homo sapiens) [44, 37, 60]. Molecular biology enters the post genomic era [77] and it is possible to decode the molecular mechanism of life for the first time in the human history.

Another common point for life on earth is that proteins play a central role in every life on earth. Protein is a complex, high-molecular-mass, organic compound that consists of amino acids joined by peptide bonds. There are twenty amino acids out there to form proteins. Proteins perform various biological functions, from structural roles to enzymes that catalyze chemical reactions, to immune responses. The sequence of amino acids in a protein is called primary structure of a protein, which is determined by the gene that

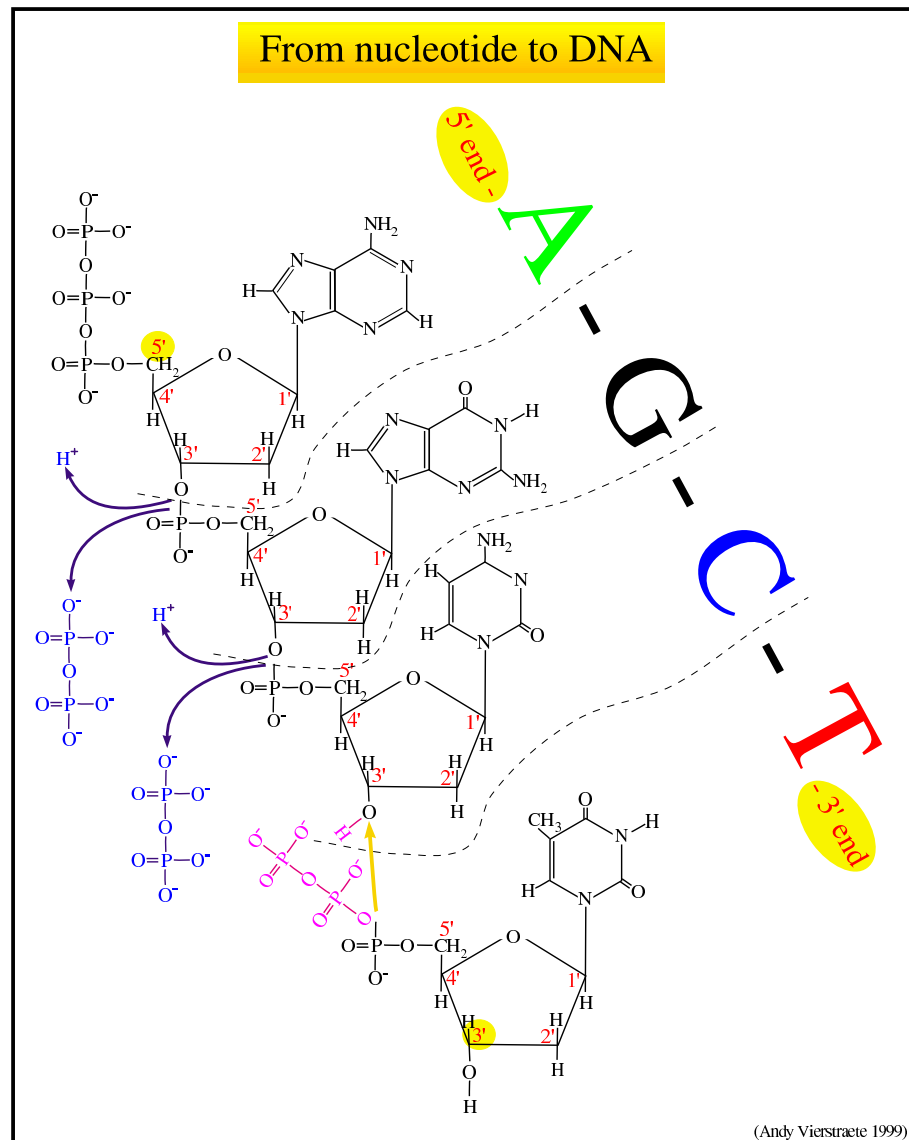


Figure 1.3.1: DNA is formed by coupling the nucleotides between the phosphate group from a nucleotide (which is positioned on the 5th C-atom of the sugar molecule) with the hydroxyl on the 3rd C-atom on the sugar molecule of the previous nucleotide [122]. A-G-C-T form the four bases of a DNA molecular.

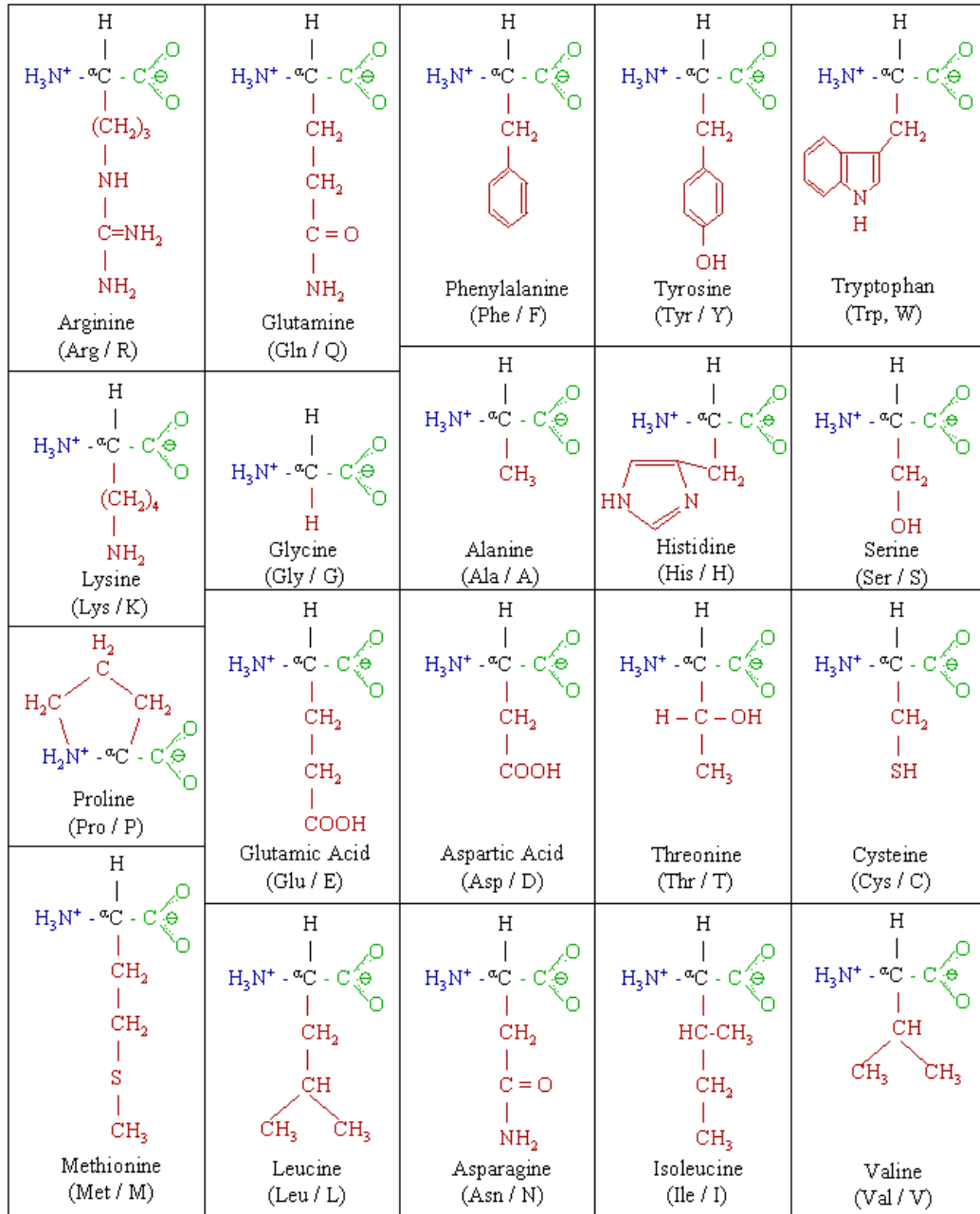


Figure 1.3.2: The table of all twenty amino acids found in proteins [3]. Their names and chemical makeups.

encodes the protein. In addition to the primary structure, a protein also demonstrates 3D structures, called secondary structures and tertiary structures. As a matter of fact, in order for a protein to function, it must be properly folded into a proper 3D shape.

The central dogma of molecular biology states: **DNA** → **RNA (Ribonucleic Acid)** → **Protein**. It means that DNA encodes messenger RNA and is transcribed into messenger RNA. Messenger RNA is then translated (after some post-transcription processing, e.g. splicing) into protein molecular. DNAs are like blueprints, recording information of how to construct proteins. On the other hand, day-to-day molecular functions are carried out by proteins. RNA molecular has similar structure as DNA, but it is single stranded and the thymine (T) base is replaced by uracil (U). The central dogma of molecular biology can be more specifically stated as follows [30]:

- The information contained in DNA is duplicated via the replication process.
- DNA directs the production of encoded messenger RNA (mRNA) through a process called transcription.
- In eukaryotic cells, the mRNA is then processed and migrates from the nucleus to the cytoplasm of the cell.
- In the final stage of the information-transfer process, messenger RNA carries the encoded information to protein-synthesizing structures called ribosomes. Through a

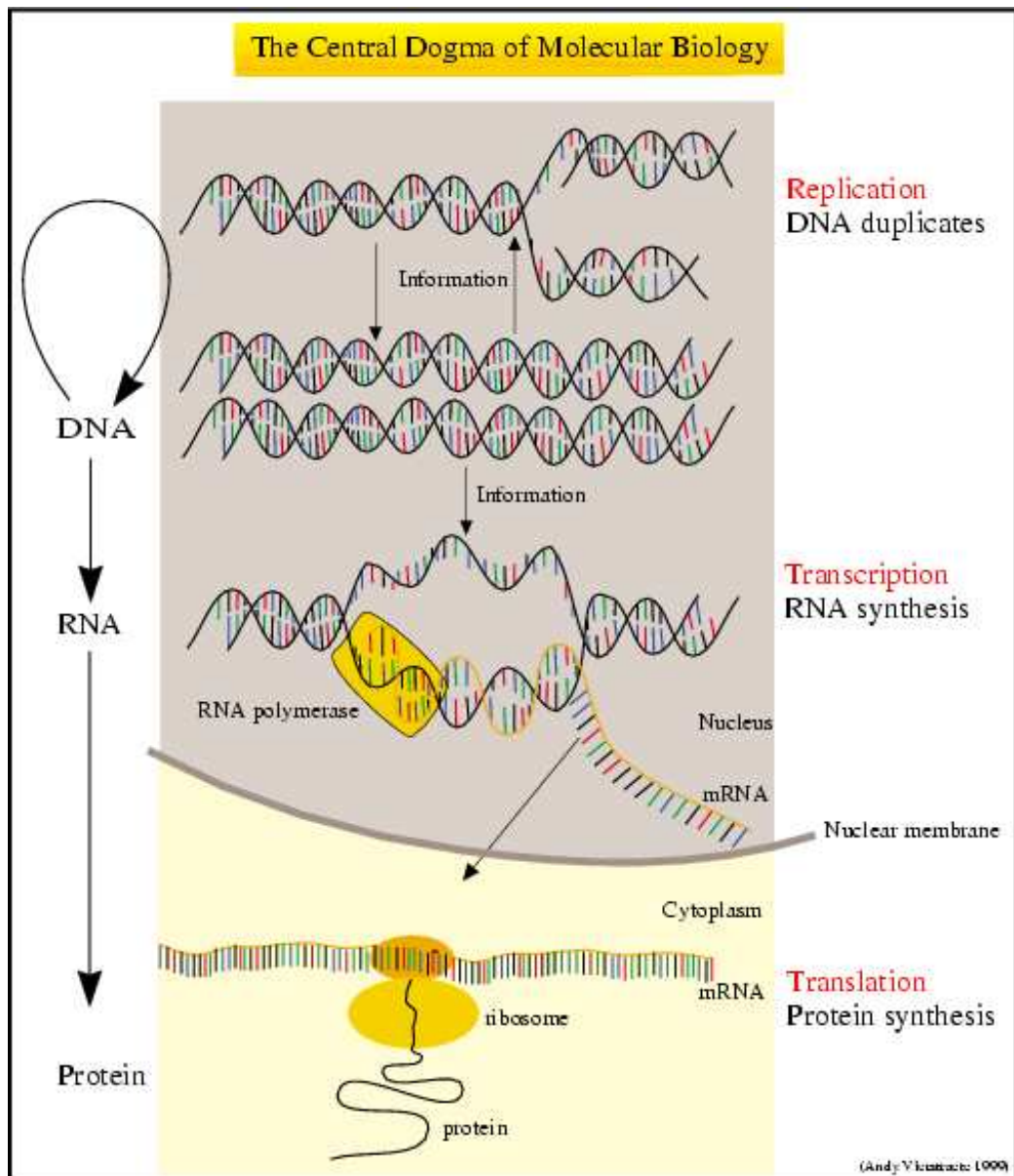


Figure 1.3.3: The central dogma of molecular biology. This figure illustrates the process of DNA replication, DNA transcription into messenger RNA and messenger RNA translation into protein [122].

process called translation, the ribosomes use this coded information to direct protein synthesis.

How does DNA encode proteins? It turns out to be quite simple, although discovery of it is not. Using the four bases of adenine (A), guanine (G), cytosine (C), and uracil (U) for RNA, every three consecutive bases, called a codon, encodes one amino acid. Given the four base types, there are 64 codes available. Yet only 20 are used, as illustrated in 1.3.4. For example GUU, GUC, GUA, GUG all encode valine (Val).

Protein is the molecular that carries out most of life's function. Although the expression level of messenger RNA is not a direct indicator of corresponding protein level due to post-transcription modifications, their expression levels can be measured relatively easily and cheaply in large quantity given currently techniques (microarray experiments). The resulting microarray expression data set is the primary data set we are dealing with in this thesis. We will take a closer look at microarray experiment in the next section.

1.3.2 Microarray Experiment

Microarray technique enables biologists to monitor expression levels of thousands of genes or ESTs simultaneously. Modern microarray technology was originated from the Southern blot (named after E. M. Southern, a British biologist), which was the first array of genetic

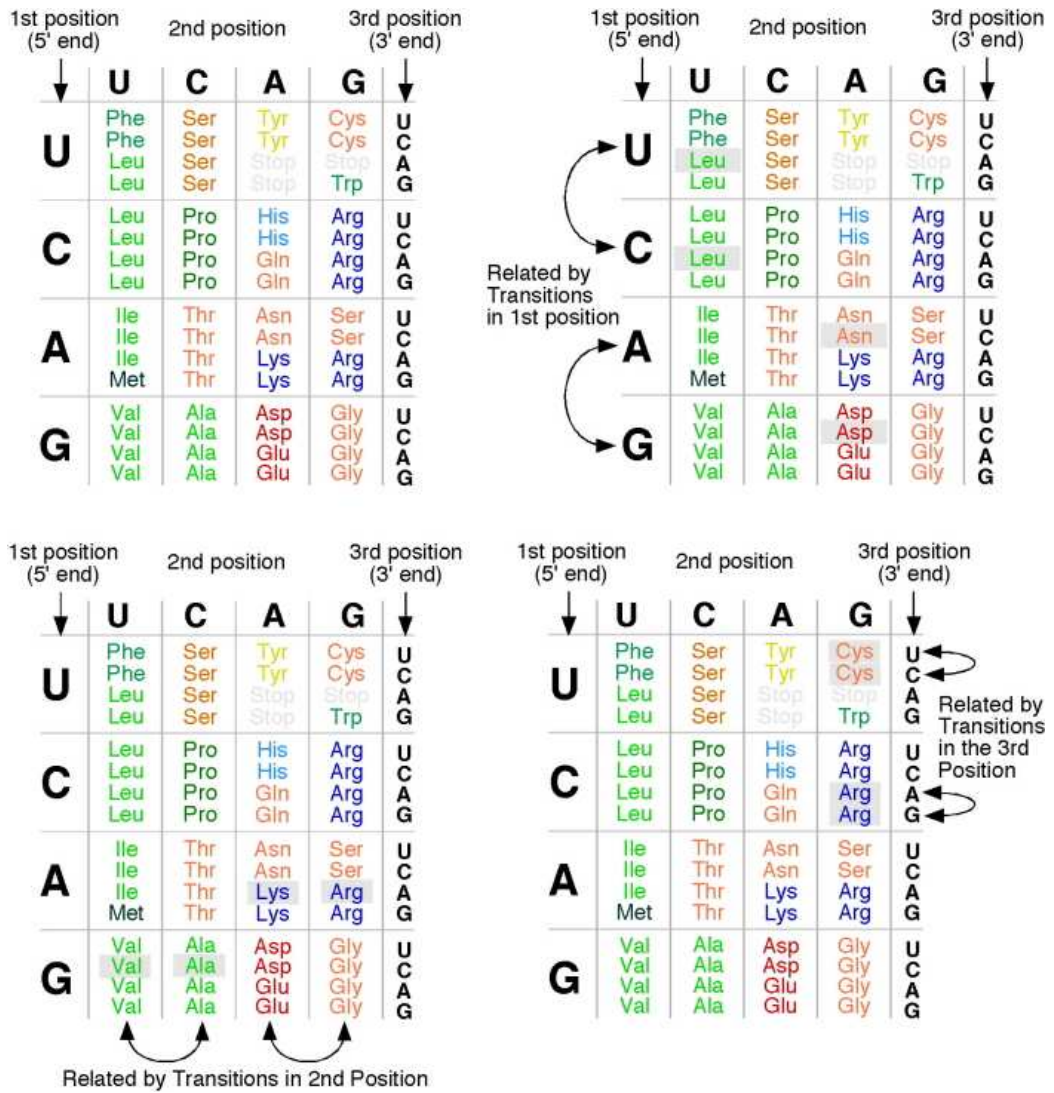


Figure 1.3.4: Genetic Coding: Encoding of amino acids using four bases (A,C,U,G) found in RNA molecular. Every three bases form a codon, which encodes one amino acid [4]. Amino acids are the building stones of proteins. The DNA T base is substituted by a RNA U base.

material. The basic principle behind these techniques is that DNA and RNA strands can be labeled for detection and then used to probe other nucleic acid molecules that have been attached to a solid surface. In the most general form, a DNA array is a chip made of nylon membrane, glass or plastic. Usually, the chip is arranged in a regular grid-like pattern and segments of DNA strands are either deposited or synthesized within individual grids. Figure 1.3.5 shows the basic principles of microarray experiment. Once the array is prepared, a microarray experiment involves three basic steps: sample preparation and labeling, sample hybridization and washing, and microarray image scanning and processing. Gene chips are well commercialized now, e.g., the first patented DNA microarray wafer chip, GeneChip from Affymetrix, Inc.

After image scanning and processing, microarray experiments normally produce a two dimensional array of numbers. In this thesis, genes are organized in the column direction and samples organized in the row direction as shown in Figure 1.3.6. Each column in such an array is the expression levels of all genes of one sample in the experiment. Each row is the expression levels of one gene across different sample tissues. Computational algorithms are used to analyze such data set, trying to discover novel correlations between genes and sample tissues.

For a more detailed account of microarray gene expression experiment, please refer to [84, 16].

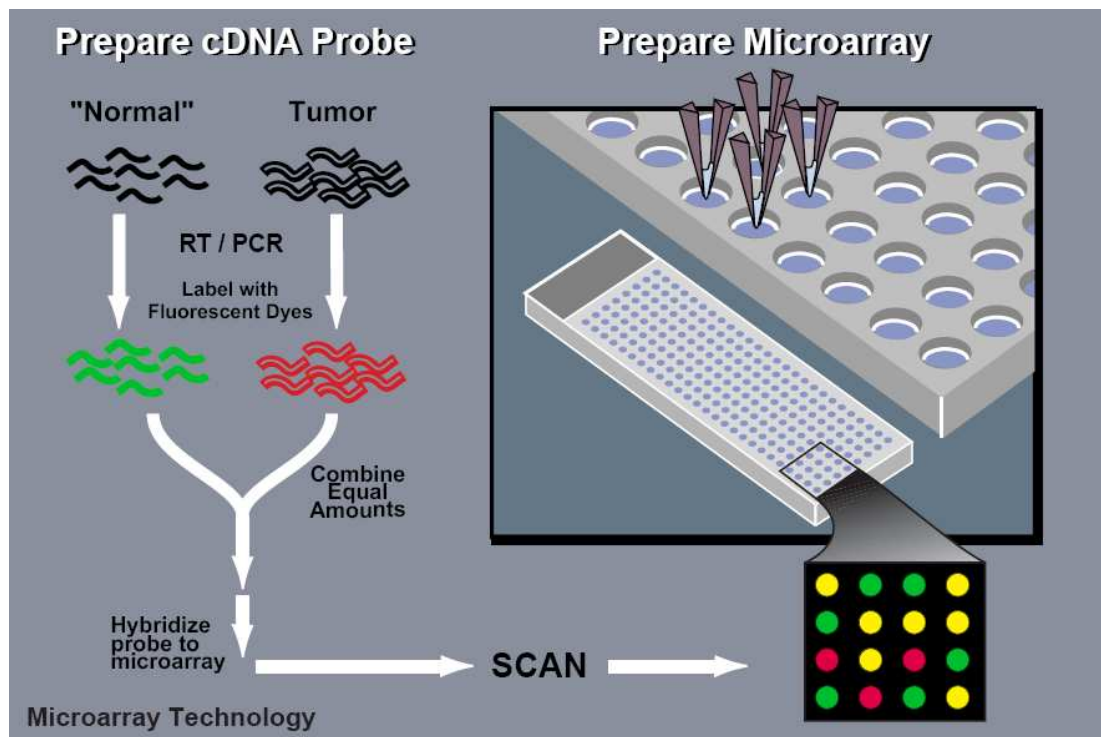


Figure 1.3.5: The workflow of cDNA microarray experiment. Tissue samples from different sample classes are processed by RT/PCR for mRNA amplification and labeled using fluorescent material. They are then exposed to cDNA chip for hybridization. Resulting chip is then scanned and processed to produce a two dimensional numerical array of microarray gene expression data set that is used by data analysis algorithms. [65]. There are other techniques to build microarrays.




	Sample 1 label 	Sample 2 label 	...	Sample M label 
Gene 1	1.23	-0.98		-0.37
Gene 2	-0.22	-0.83		0.53
.			...	
.				
.				
Gene N	4.48	-5.73		-2.39

Figure 1.3.6: The microarray data set after processing. It is a two dimensional numerical array with genes as rows and tissue samples as columns. Samples are labeled by some external labels, such as normal or cancer.

1.3.3 Gene Ontology and Gene Annotations

Gene ontology (GO) [28] is a shared structured library of biological terms. Currently GO is divided into three distinct parts: molecular function, biological process and cellular component. Genes and their products in different organisms are annotated using this common terminology by GO collaborators. GO annotations represent a large repository of biological knowledge that is accessible to computational algorithms. SOURCE [32] from Stanford is a good online database for querying available annotations on several model organisms. In this thesis, we propose to use such biological knowledge database to facilitate gene selection.

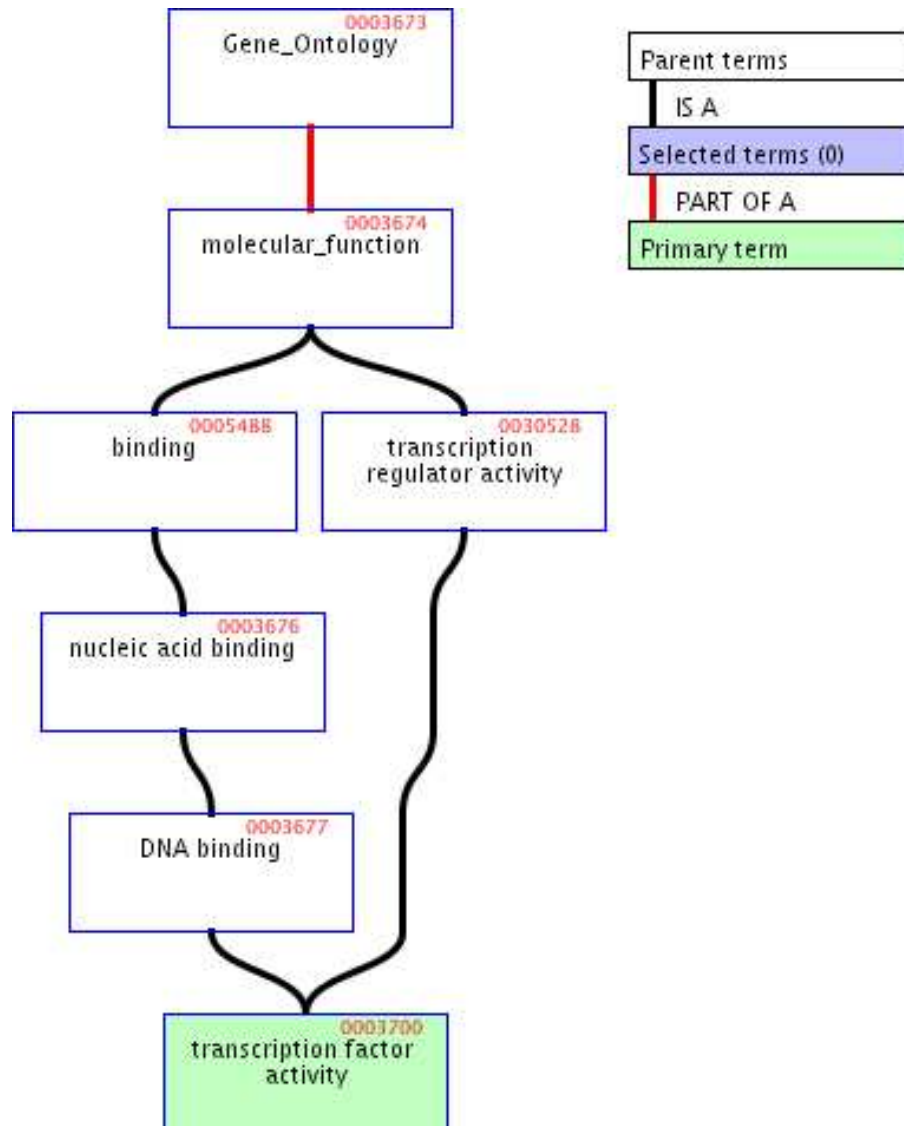


Figure 1.3.7: The GO hierarchy. The figure shows all GO terms leading to GO term GO:0003700, or “transcription factor activity”. This GO term lies in the molecular function branch of gene ontology. GO terms form a DAG (directed acyclic graph). There are two paths leading from root GO node to GO:0003700, either from the GO term “binding” or through the GO term “transcription regulator activity” [96].

Gene ontology consists of a set of shared biological terms and relationships such as “is-part-of” and “is-a” between those terms. GO terms and their relationships form a DAG (directed acyclic graph). Each GO term could have several “parent” terms and several “child” terms. However, no cycle is allowed. GO can be downloaded from [27] in various format and can be queried easily using [9, 96]. Figure 1.3.7 shows the hierarchical of part of the ontology involving GO term “transcription factor activity”. Figure 1.3.8 shows the annotated genes and gene products by different biological database groups for GO term “calcium transporting ATPase activity”.

It is shown in the literature that gene ontology-based similarities between genes carry significant information of the functional relationships [11]. Gene ontology has been incorporated into several microarray data analysis and visualization algorithms/tools for various purposes, in the context of cluster validation [17], visualization of distribution of some scores over GO terms [25]. Authors in [11, 124] concluded that the GO-driven similarity and expression correlation are significantly interrelated. Another avenue of research focuses on detecting over-represented GO terms in a set of co-expressed genes [7]. Tuikkala etc. used Gene Ontology to find relevant genes for estimating missing expression values [117]. Huang [64] proposed to integrate known gene functions into a gene distance metric and used it in the context of gene clustering.

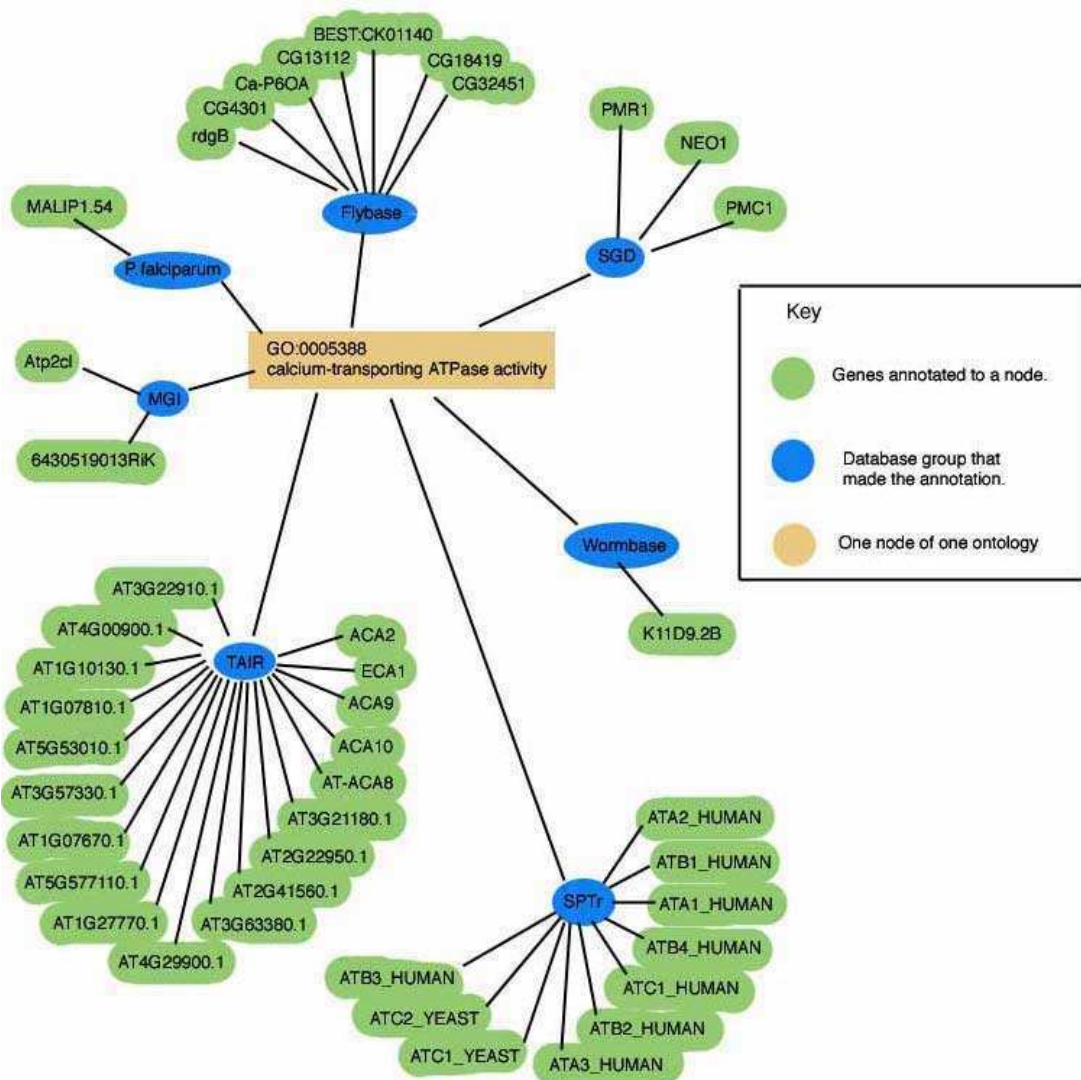


Figure 1.3.8: The GO annotations are provided by participating databases [27], such as Flybase [45] and SGD [105]. This figure illustrates the annotations of various genes and gene products using one GO term GO:0005388, or “calcium transporting ATPase activity”, by different biological database groups.

1.4 Gene Selection for Microarray Experiments

In this section, we formalize the problem we are trying to solve. A formal formulation of our problem will be discussed in the first section. We then discuss the limitations of current gene selection algorithms and our proposed improvements.

1.4.1 Formulation of Our Problem

In this section we formalize the problem of gene selection for microarray data sets.

Let \mathcal{R} be the set of real numbers and \mathcal{N} be the set of natural numbers. Let $\mathcal{G} = \{g_1, g_2, \dots, g_n\}$ be the set of all genes that are used in one study, $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ be the set of all experiments performed, $\mathcal{L} = \{l_1, l_2, \dots, l_l\}$ be the set of sample class labels of interest. We assume $\mathcal{G}, \mathcal{S}, \mathcal{L}$ are fixed for any given study. Let $n = |\mathcal{G}|$ be the total number of genes, $m = |\mathcal{S}|$ be the total number of experiments and $l = |\mathcal{L}|$ be the total number of class labels. A microarray expression data set used in our study can be defined as $\mathcal{E} = (\mathcal{G}, \mathcal{S}, \mathcal{L}, L, E)$, where L is a function $\mathcal{S} \rightarrow \mathcal{L}$ such that for $s \in \mathcal{S}$, $L(s) \in \mathcal{L}$ is the class label for sample s ; E is a function $\mathcal{G} \times \mathcal{S} \rightarrow \mathcal{R}$. For $g \in \mathcal{G}$ and $s \in \mathcal{S}$, $E(g, s)$ is the expression level of gene g in experiment s . In the bioinformatics community, the function E is normally presented as a two dimensional array of real numbers.

Sometimes we need to treat the set of samples \mathcal{S} as a multiset (or bag). In this case we refer to the set of samples as $\mathcal{S}^1 = \{(s_1, 1), (s_2, 1), \dots, (s_m, 1)\}$. More formally, \mathcal{S}^1 is defined as a pair $(\mathcal{S}, \mathcal{M}_1)$, where \mathcal{M}_1 is a function that is always 1 ($\mathcal{M}_1(s) = 1, \forall s \in \mathcal{S}$). A multiset is a set that allows duplication. In the case of \mathcal{S}^1 , \mathcal{S} is the underlying set and \mathcal{M}_1 is the multiplicity function for elements in the underlying set. We refer to multiset $(\mathcal{S}, \mathcal{M})$ as the bootstrap sample set, where \mathcal{M} is an arbitrary function $\mathcal{S} \rightarrow \mathcal{N}$. We will discuss the bootstrap sample set in detail later in Chapter 2.

For simplicity of presentation, we use a subscripting scheme to refer to elements in \mathcal{E} . Let $\mathcal{E}(G, \mathcal{S}) = (G, \mathcal{S}, L(S), L, E)$ where $G \subseteq \mathcal{G}$ and \mathcal{S} is a bootstrap sample set $(\mathcal{S}, \mathcal{M})$. We further use $L(\mathcal{S})$ to denote the set of class labels for the set of experiments \mathcal{S} . Since G is a subset of \mathcal{G} and the underlying set of \mathcal{S} is \mathcal{S} , functions L and E are well defined for $\mathcal{E}(G, \mathcal{S})$.

The task of sample classification is to design a classifier $\Psi(D)$ that predicts the class label c of a new experiment instance $D = (d_1, d_2, \dots, d_n)$. D is a vector of expression levels of n genes. Gene selection is performed before sample classification. Suppose n' number of genes are to be selected. Let Φ be a function mapping from feature sets \mathcal{G} to G' , $G' \subset \mathcal{G}$ and $|G'| = n'$. The goal of feature selection is to find the best mapping Φ_0 such that classifiers trained using $\mathcal{E}(\Phi_0(\mathcal{G}), \mathcal{S})$ perform best on unknown testing samples. In the case of single-feature based feature selection, a scoring function $F(\mathcal{E}(\{g\}, \mathcal{S}))$ is defined, $g \in \mathcal{G}$. Since the second parameter stays same for all $g \in \mathcal{G}$, we also write the scoring function simply as

$F(g)$. Without loss of generality, we assume F to be non-negative and the larger F score is, the more discriminative a gene is.

Gene ontology [28] is a controlled biological vocabulary, which specifies biological terms as well as relationship among them. Currently relationships between GO terms form a DAG (directional acyclic graph) [29]. GO annotations are produced by GO collaborators tagging gene products of their biological functions with GO terms. We refer to the whole set of GO terms as $GO = \{go\}$ where go is an individual GO term. Given two GO terms go_k and go_l , we use $go_k \Rightarrow go_l$ to represent the fact that there exists a path from go_k to go_l in some GO hierarchy. When two GO terms have directly parent-child relationship in GO hierarchy, we denote it using $go_k \rightarrow go_l$. We further use notation $g \in go$ to emphasize the fact that gene g is annotated with GO term go . Following this notation, we sometimes treat a GO term as a set of genes that are annotated with the corresponding GO term. Particularly, we use notation $|go|$ to represent the number of genes that are annotated with GO term go . Gene annotations are assumed transitive in this work, in the sense that if $g \in go_k$, $\forall go_l \Rightarrow go_k, g \in go_l$.

1.4.2 The Limitations of Existing Gene Selection Algorithms and Our Proposed Approaches

Current research in feature subset selection in the microarray expression data analysis generally ignores the relationships between genes deliberately. However, the assumption of independence between genes over simplifies the complex relationship between genes in biological systems. Genes are well known to interact with each other through gene regulatory networks. As a matter of fact, the common assumption of popular cluster analysis on microarray expression data sets [70] is that co-regulated genes have similar expression profiles. Bø [15] proposed to calculate discriminant scores for a pair of genes instead of each individual gene. We showed in earlier research that the concept of virtual gene (correlations between genes) [134] could help improve gene selection. A virtual gene is a linear combination of real genes and the expression level of a virtual gene is calculated from the linear combination of expression levels of constituent real genes. Pairwise virtual gene is examined in [134]. Better sample classification performance is obtained using these virtual genes instead of real genes. This topic is covered in more details in Chapter 3. Further in Chapter 5 we integrated GO annotation and extended the concept of virtual gene to arbitrary sized gene groups (normally small biologically related gene groups). In Chapter 2, we propose a novel gene selection algorithm called BFSS (Boost Feature Subset Selection)

based on permutation analysis. BFSS aims to broaden the variety of selected genes. It selects subsequential genes based on the set of genes that has been selected so far and focuses on previously difficult samples.

Also ignored by existing gene selection algorithms is the domain knowledge available about genes and their products. Microarray gene expression data set reveals only part of the overall biological process. The integration of other data sets, such as GO annotations of genes and their products, with microarray gene expression data set could be proven useful. In Chapter 5, a novel gene extraction algorithm is proposed by integrating gene annotations with microarray expression data set for the purpose of sample classification. The sample classification problem on microarray data set is fraught with the false positive problem due to the skewed dimensionality. Gene annotations can be used to alleviate the false positive problem to some extent as we demonstrate in Chapter 4.

1.5 Organization of This Thesis

This thesis is divided into six chapters. Chapter 1 introduces background both in the computational science and the biological science surrounding the work in this thesis. Chapter 2 introduces a novel meta feature selection algorithm based on permutation analysis [135, 136]. Chapter 3 introduces pairwise virtual gene algorithm [133, 134]. Virtual gene

algorithm is a gene extraction algorithm that investigates correlation between genes for the purpose of sample classification. From Chapter 4, we introduce domain knowledge into the process of sample classification on microarray expression data set, by integrating GO annotations of genes and their products. In Chapter 4, gene annotations are used to alleviate the false positive rate problem in sample classification using microarray expression data sets [132]. In Chapter 5, virtual gene algorithm is extended to investigate gene correlations beyond pairs using gene annotations. We conclude this thesis in Chapter 6.

Chapter 2

BFSS: Boost Feature Subset Selection

As discussed in Chapter 1, one class of typical approach of feature selection is to calculate some discriminative score using data associated with a single feature. Such discriminative scores are then sorted and top ranked features are selected for further analysis. In the bioinformatics field, such single-feature based feature subset selection algorithms are widely used to select informative genes from the microarray expression data set. However, such an approach will result in redundant feature set since it ignores the complex relationships between features (genes). Recent researches in feature subset selection began to tackle this problem by limiting the correlations in the selected feature subset, also shown in Chapter 1 in the section of redundancy based feature selection algorithms. In this chapter, we propose a novel general framework called BFSS (Boost Feature Subset Selection) to improve the

performance of single-feature based discriminative scores using bootstrapping techniques. Features are selected from dynamically adjusted bootstraps of the training data set instead of the training data set itself. We tested our algorithm on three well-known publicly available microarray expression data sets in bioinformatics community. Encouraging results are reported here.

Closely related to our methodology is the bootstrap techniques used in ensemble classifier design. Two popular algorithms: bagging [20] and boosting [47, 82] are proposed to train individual classifiers using independently sampled training sets. The final classifier is a majority vote of all individual classifiers. Bagging and boosting differ on how samples are drawn from the training set for each constituent classifier. In bagging, the sampling of the training set for each classifier is independent of each other. However, in the case of boosting, which training samples are chosen for subsequent classifiers are based on the performance of the current classifier. The sampling probabilities of those training samples that are misclassified by current classifier are increased. In this way, subsequent classifiers focus more on misclassified training samples. It is worth mentioning that boosting has its roots in PAC (probably approximately correct) learners [101] and it is also proven theoretically that learners, each performing only slightly better than random, can be combined to form arbitrarily good hypothesis (zero prediction error on the training set) [72, 85].

This chapter is organized as follows. An illustrating example is given in Section 1. Our

BFSS algorithm is detailed in Section 2. Extensive experiment is performed and reported in Section 3. We conclude this chapter in section 4.

2.1 A Motivating Example

Single-gene based discriminative scores, although simple, are widely used in gene selection. It is interesting to see those algorithms that are traditionally regarded as less capable working relatively well on microarray expression data sets. It is probably due to the skewed dimensionality of microarray expression data set, where the assumption of traditional feature subset selection breaks down.

The most obvious drawback of single-gene based gene selection algorithms is the fact that those algorithms ignore the relationships that exist between genes. There is no guarantee that the combination of two “good” features will necessarily produce a “better” classifier. As an illustrating synthetic example in Figure 2.1.1, the expression levels of three genes across 100 samples are plotted. Samples are labeled using two class labels: either cancer (grey background) or normal (white background).

The first two genes, gene 1 and gene 2 behave similarly. In majority of the samples (samples 1 to 40 and samples 61 to 100, or 80% of samples), the expression levels of gene 1 or gene 2 can be used to predict sample class labels effectively. Actually the expression levels

Synthetic Example of Expression Levels of Three Genes

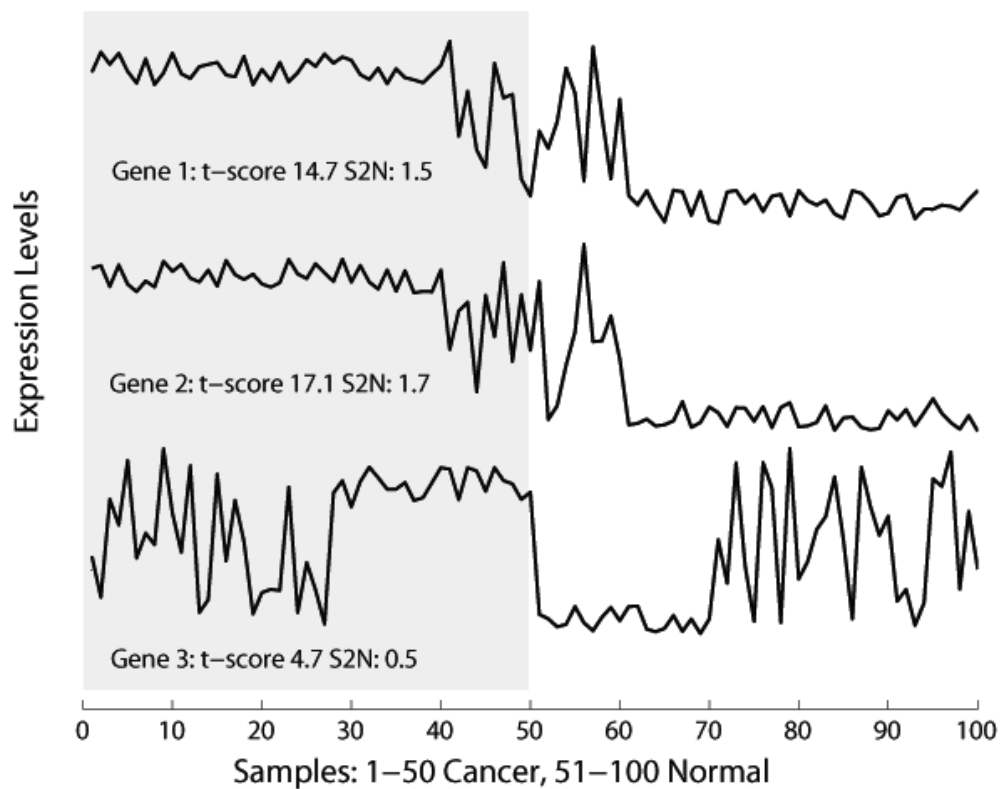


Figure 2.1.1: An illustrating example for the BFSS algorithm: redundancy in a selected gene set. Genes with high individual discriminative scores may not be overall good choices for a gene subset. Gene 1 and Gene 2 have higher individual discriminative scores, yet their expression levels are highly correlated across different experimental samples. Gene 3 on the other hand provides new information, although its discriminative score is lower.

of these two genes are generally higher in cancer samples than in normal ones. However, the expression levels of these two genes in samples 41 to 60 (20% of samples) are more mixed across cancer/normal class distinction.

Gene 1 and gene 2 score high in terms of t-score and S2N scores as shown in Figure 2.1.1. Gene 3 is obviously a less capable predictor when considered alone, compared to gene 1 and gene 2. Clear trend exists in the expression levels of gene 3 in samples 31 to sample 70 (40% of samples). However it varies across cancer/normal labels in the rest samples (60%). Gene 3 scores much lower than gene 1 and gene 2 in terms of t-score and S2N as expected. Using t-score and S2N, we can rank these three genes based on their salience in predicting cancer/normal class labels as: gene 2 > gene 1 > gene 3.

However, t-score and S2N do not consider the fact that gene 1 and gene 2 behave similarly. They both work well in samples 1 to 40 and samples 61 to 100. They both share more difficult samples, namely samples 41 to 60. If two genes out of the three genes are to be chosen for further data analysis tasks, would it be wise to use both gene 1 and gene 2, as suggested by their relatively high single-gene based discriminative score (t-score and S2N) rankings? This is the very problem we are addressing in this chapter. We empirically show later that it is not the case. Choosing gene 2 and gene 3 might be a better idea as gene 3 “covers” the more difficult samples that gene 2 fails to cover. Our BFSS feature selection algorithm works to focus subsequent feature selection on those more difficult samples that

previously selected genes failed.

2.2 BFSS: Boost Feature Subset Selection

In this section we formulate our boost feature subset selection algorithm (BFSS). It is worth noting that our BFSS algorithm is a general feature subset selection algorithm although our application focuses on gene selection. In this chapter, the word “feature(s)” and “gene(s)” are used interchangeably in this context.

2.2.1 BFSS: Boost Feature Subset Selection Algorithm

In this section, we elaborate our new algorithm. First we will define some concepts and then describe our BFSS algorithm. Several concepts are widely used elsewhere. For example, bootstrapping as a method for estimating the sampling distribution of an estimator by resampling with replacement from the original data set, is well defined in the statistics community. We adapt these concepts using consistent notation defined in this chapter.

A bootstrap sample set $\mathcal{S}_b = (\mathcal{S}, \mathcal{M})$ is a multiset of samples randomly drawn with replacement from the original set of samples \mathcal{S} . $\mathcal{M}(s), s \in \mathcal{S}$, is the multiplicity of item s . As a result, the same sample $s \in \mathcal{S}$ can appear more than once or does not appear at all in \mathcal{S}_b .

The cardinality of \mathcal{S}_b is denoted by m_b . The sampling probability of each sample in \mathcal{S} is determined by a probability table p . For each $s \in \mathcal{S}$, p_s is the sampling probability. This probability table is updated during the feature selection process in our algorithm.

Definition 2.2.1 A bootstrap sample set $\mathcal{S}_b = (\mathcal{S}, \mathcal{M})$ of size m_b is a multiset of samples resulting from random sampling from \mathcal{S} with replacement. The probability of each sample $s \in \mathcal{S}$ being sampled is p_s .

Definition 2.2.2 A bootstrap \mathcal{B} of a training data set $\mathcal{E} = (\mathcal{G}, \mathcal{S}, \mathcal{L}, L, E)$ using bootstrap sample set \mathcal{S}_b is a data set defined as

$$\mathcal{B} = (\mathcal{G}, \mathcal{S}_b, \mathcal{L}, L, E) \quad (2.2.1)$$

Definition 2.2.3 The worst set of samples \mathcal{S}_{worst} of size δ with respect to bootstrap data set $\mathcal{E}(g, \mathcal{S}_b)$ and a single-gene based scoring function F is defined as a multiset:

$$\underset{S \subseteq \mathcal{S}_b \text{ and } |S|=\delta}{\operatorname{argmax}} (F(g, \mathcal{S}_b - S)) \quad (2.2.2)$$

Here $\mathcal{S}_b - S$ is a set by removing S from \mathcal{S}_b . We also call $\mathcal{S}_b - \mathcal{S}_{worst}$ the **best set of samples**.

A bootstrap of training set is defined using the definition of a bootstrap sample set. A bootstrap \mathcal{B} is a new data set and is also defined using the five-tuple notation we used to define the microarray expression data set. The only difference is that the second element is a bootstrap sample set. A bootstrap \mathcal{B} of a microarray expression data set is defined as

$(\mathcal{G}, \mathcal{S}_b, \mathcal{L}, L, E)$. \mathcal{B} shares the same set of genes \mathcal{G} , same set of class labels \mathcal{L} , same sample class label mapping L , and same expression levels mapping E with \mathcal{E} .

Given a bootstrap \mathcal{B} , a gene g and a score function F , the worst set of sample of size δ is a set of samples such that by removing them from the data set \mathcal{B} , best F score for gene g is achieved. We refer to all other samples in \mathcal{S}_b other than those in the worst set of samples the best set of samples. Both the worst set of samples and the best set of samples are multisets.

Algorithm 1 *WorstSampleSet* : Calculate the **worst set of samples** using a greedy algorithm

Require: $\mathcal{E} = (\{g\}, \mathcal{S}, L(\mathcal{S}), L, E)$, F as a single-gene based discriminative score

Require: δ is the size of worst set of samples

Ensure: S' is the **worst set of samples** with respect to \mathcal{E} and F

- 1: S', S_0 to be empty sets
 - 2: **for all** $s \in \mathcal{S}$ **do**
 - 3: $S_1 \leftarrow \mathcal{S} - \{s\}$
 - 4: calculate $F(\mathcal{E}(\{g\}, S_1))$, add score to S_0
 - 5: **end for**
 - 6: sort S_0 , add samples s corresponding to top δ scores in S_0 to S'
 - 7: **return** S'
-

By the definition of the worst set of samples with respect to gene g and score F , it is NP hard to find such set of samples since the power set of samples needs to be examined. We employ a simply greedy algorithm as described in Algorithm 1. For each sample $s \in \mathcal{S}_b$, F scores for gene g on each sample set $\mathcal{S}_b - \{s\}$ are computed. Such scores are ranked and the samples corresponding to the best δ scores are treated as the worst set of samples. δ is an input parameter of our algorithm. However, as shown later in this section, fixed value of δ is used for all data sets we tested with good results.

Algorithm 2 BFSS : Boost Feature Subset Selection

Require: $\mathcal{E} = (G, \mathcal{S}, L(\mathcal{S}), L, E)$; n' as the number of genes to be selected; F as a single-gene based discriminative score

Ensure: G' as the selected gene set by BFSS using F .

- 1: Let p to be a vector of length m . Set initial value of p to be $1/m$ (m is the number of samples in \mathcal{E}). Set G' as an empty set.
 - 2: $\mathcal{E}' \leftarrow \mathcal{E}$
 - 3: **for** $|G'| < n'$ **do**
 - 4: generate bootstrap sample set \mathcal{S}_b and bootstrap \mathcal{B} of training set \mathcal{E}' by random sampling using probabilities p
 - 5: calculate score F on bootstrap \mathcal{B} , refer to this score as F' , keep track of the best score so far
 - 6: add top ranked gene g based on F' to G'
 - 7: find worst δ samples \mathcal{S}_{worst} based on $\mathcal{E}(g, \mathcal{S}_b)$ using Algorithm 1
 - 8: reduce p_s where $s \in \mathcal{S}_b - \mathcal{S}_{worst}$ by a factor of ϵ and normalize p so that it represents a distribution
 - 9: remove g from \mathcal{E}'
 - 10: **end for**
 - 11: **return** G'
-

Boost feature subset selection algorithm (BFSS) is shown in Algorithm 2. After some initialization, the algorithm first generates a bootstrap \mathcal{B} of training set \mathcal{E} . This involves the generation of a bootstrap sample set of size m_b and then the bootstrap itself. We used the function `sample()` provided in R [52] environment to generate the bootstrap sample set. It is a sampling function with replacement using specific probability p_s for each observations in \mathcal{S} . Bootstrap \mathcal{B} is then generated by some subscripting using the generated bootstrap sample set and training set \mathcal{E} .

After bootstrap \mathcal{B} of a training set is generated, the F score is then calculated for each gene in \mathcal{B} . Best F score so far is kept during the computation, so is the gene associated

with it. In the next step, the gene with best F score for current bootstrap B is selected and added to the selected gene set. Based on the selected gene, BFSS then identifies the worst set of samples with respect to the currently selected gene and the single-gene based scoring function using Algorithm 1. The probability table p for generating bootstraps is modified by reducing the probabilities for the best set of samples by a constant factor. The probability of those good samples being selected in subsequent analysis is thus reduced, focusing BFSS onto those samples that previously selected genes would not perform well. The currently selected gene is then marked as selected and not considered further by the BFSS algorithm. BFSS repeats this process until n' genes are selected.

There are three parameters m_b (size of bootstrap sample set), δ (size of worst set of samples), and ϵ (the degradation factor of sampling probability) used in our algorithm. We experimentally chose δ to be 0.96 of the number of training samples in a data set and ϵ to be 0.96. We set m_b to be twice the number of samples in the training set so that a bootstrap is more representative. After fixing these three parameters, there is virtually no more need of tweaking our BFSS algorithm. We used these same parameters for all the three data sets we experimented with and achieved good performance on most of them. This indicates BFSS's good property of requiring little tuning for different data sets.

2.2.2 Computational Complexity of BFSS

Using notation defined previously, assume n' number of genes are to be selected. Further assume the computational complexity of single-gene based discriminative score F to be $\Theta_F(m)$. As defined earlier, m_b is the size of the bootstrap sample set and m is the number of samples in the training set.

First consider the *WorstSampleSet* algorithm (Algorithm 1). For each sample in a bootstrap, we need to calculate F score. This will cost $O(m_b \times \Theta_F(m_b - 1))$. If the selection of δ best values is implemented using sorting, it will cost $O(\delta \times m_b \log m_b)$. Overall *WorstSampleSet* requires $O(m_b \times (\Theta_F(m_b) + \delta \times \log m_b))$.

The BFSS algorithm has main loop at line 3 that will repeat n' times for each of the genes to be selected. Within the loop body, line 4 generates bootstrap sample set, which can be computed in $O(m_b \times \log m)$. Line 5 computes F score for each gene on bootstrap \mathcal{B} , which requires $O(n \times \Theta_F(m_b))$. The top ranked gene can be identified in constant time (line 6) if we keep track of it in the previous step. Line 7 requires $O(m_b \times (\Theta_F(m_b) + \delta \times \log m_b))$ as analyzed in the previous paragraph. Line 8 requires $O(m)$ time to degrade the probability vector properly. And line 9 requires constant time to remove selected gene from training data set. It is obvious that the computational cost is dominated by line 5 when single-gene based discriminative score F for all genes on bootstrap \mathcal{B} is computed.

Overall, BFSS have time complexity of $O(n' \times n \times \Theta_F(m_b))$, compared to original time complexity of F to be $O(n \times (\log n + \Theta_F(m_b)))$. Since it is generally very fast to compute function F and n' is usually small, the computation of BFSS can be completed in reasonable time.

Space required by BFSS algorithm is marginal compared to the size of input training data set \mathcal{E} . Most space needed is for storing bootstraps \mathcal{B} . We omit detailed analysis here.

2.3 Experiments

We performed extensive experiments on three publicly available microarray expression data sets. This section is organized as follows. In the first subsection, we briefly describe the data sets we used and the preprocessing we did on those data sets. In the second subsection, we describe how the performance of the feature selection algorithm is measured in our experiments. Then in the rest of this section, detailed experimental results are reported on each of the data set we used.

2.3.1 Data Sets and Data Preparation

Three publicly available data sets: colon cancer [8], leukemia [53] and multi-class cancer [97] are used in this study.

The first data set was published by Alon etc. [8] in 1999. It contains measurements of expression levels of 2000 genes over 62 samples, 40 samples were from colon cancer patients and the other 22 samples were from normal tissues. The minimum value in this data set is 5.8163, thus no thresholding is done. We perform base 10 logarithmic transformation and then for each gene, subtract mean and divide by standard deviation. We will refer to this data set as colon cancer data set in the rest of the chapter.

The second data set was published by Golub etc. [53] in 1999. It consists of 72 samples, of which 47 samples were acute lymphoblastic leukemia (ALL) and rest 25 samples were acute myeloid leukemia (AML). 7129 genes were monitored in their study. This data set contains a lot of negative intensity values. We use the following steps (similar to Dudoit etc.[36]) to preprocess the data set before feeding to our algorithm. First we threshold the data set with floor of 1 and ceiling of 16000. Then we filter out genes with $max/min \leq 5$ or $(max - min) \leq 500$, where max and min are the maximum and minimum expression values of a gene. After these two steps, the resulting 3927 genes are transformed using base 10 logarithmic and then the expression levels for each gene are normalized. We will refer

to this data set as Leukemia data set in the rest of this chapter.

The last data set used in this chapter is a multi-class cancer data set. Ramaswamy etc. [97] in 2001 reported study of oligonucleotide microarray gene expression involving 218 tumor samples spanning 14 common tumor types and 90 normal tissue samples. The expression levels of 16063 genes and expressed sequence tags were monitored in their experiments. The author separated the tumor samples into training set (144 samples) and testing set (54 samples). The rest 20 samples are poorly differentiated adenocarcinomas, which we do not include in our study. The training tumor set of 144 samples and 90 normal tissue samples are combined together for our study. As a data preprocessing step, we apply a thresholding of 1 and filter out genes with $max/min \leq 5$ or $(max - min) \leq 500$. The resulting data set has 11985 genes. Logarithmic transformation and normalization are then performed before data are fed to gene selection algorithms. The original data set consists of 15 class labels (14 cancers and normal). For our experiments, we only use the cancer/normal distinction. We refer this data set as multi-class cancer data set in the rest of our chapter.

2.3.2 Feature Selection and Classification Algorithms

Performance of classifiers is used as a measure of the performance of feature subset selection algorithms. In order not to be biased on which classifiers we use, three very different

general purpose classifiers are used: DLD (diagonal linear discriminant) [83], KNN (k-nearest neighbor, $k=3$) and SVM (support vector machine, with radial kernel) [23]. They represent a linear classifier (DLD), a nonlinear classifier (KNN) and arguably the most popular general purpose classifier (SVM). Since these three classifiers are fundamentally different, the differences we observed in classification performance can be properly attributed to the different feature subset selection algorithms. Cross-validation procedure is used to estimate classification performance. We use a 2-fold cross-validation rather than the commonly used 10-fold cross-validation since results generated by 10-fold cross-validation varied too much in our case. The average performance of 100 runs is reported here as the classification performance.

We systematically examine the performance of different feature subset selection algorithms using these three classifiers with different number of genes selected. The number of genes selected ranges from 2 to 100 in the increment of 2.

This chapter focuses on improving performance of single-feature based feature subset selection algorithms. Although our algorithm works for most if not all single-feature based feature subset selection algorithms, two simple widely used algorithms are used for testing: t-score and S2N. We plug these two algorithms into our BFSS algorithm and refer to the resulting new algorithms as “boost t-score” and “boost S2N” respectively. We also include the performance of our previously proposed gene selection algorithm called virtual gene

Table 2.3.1: Classification accuracy (%) without FSS on three publicly available data sets: colon cancer, leukemia and multi-class.

<i>Classifier</i>	<i>Alon</i>	<i>Golub</i>	<i>Multi-Class</i>
DLD	0.647	0.653	0.772
KNN (k=3)	0.759	0.931	0.856
SVM (Radial)	0.713	0.920	0.834

[134] in most experiments. Virtual gene algorithm is the topic of Chapter 3. Performance of all five FSS algorithms on all three data sets using all three classifiers are reported here in the following subsections.

Before testing the performance of feature subset selection algorithms, Table 2.3.1 shows the performance of the three classifiers on three data sets using all original features (after preprocessing).

2.3.3 Experiments on Colon Cancer Data Set

As shown in Figures 2.3.1-2.3.3, boosted versions of single-gene based feature subset selection algorithms outperform their original counterparts in all cases we tested. The performance of gene selection generally increases as the number of selected genes increases. There is an exception to this for DLD classifier, in which the performance of gene selection peaks when around 16 genes are selected. The performance of t-score and S2N then begin to decrease as more genes are selected. When more than 4 genes are selected, the boosted version of t-score and S2N score outperform their original counterparts. The difference is

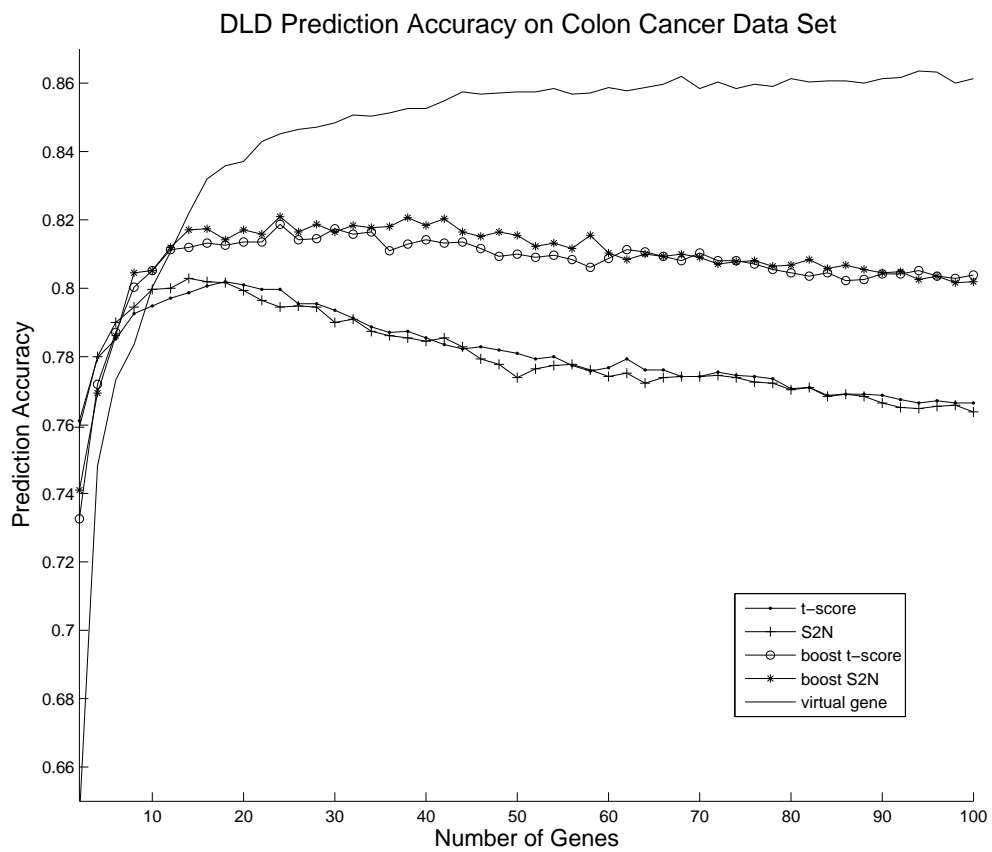


Figure 2.3.1: DLD prediction accuracy on colon cancer data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.

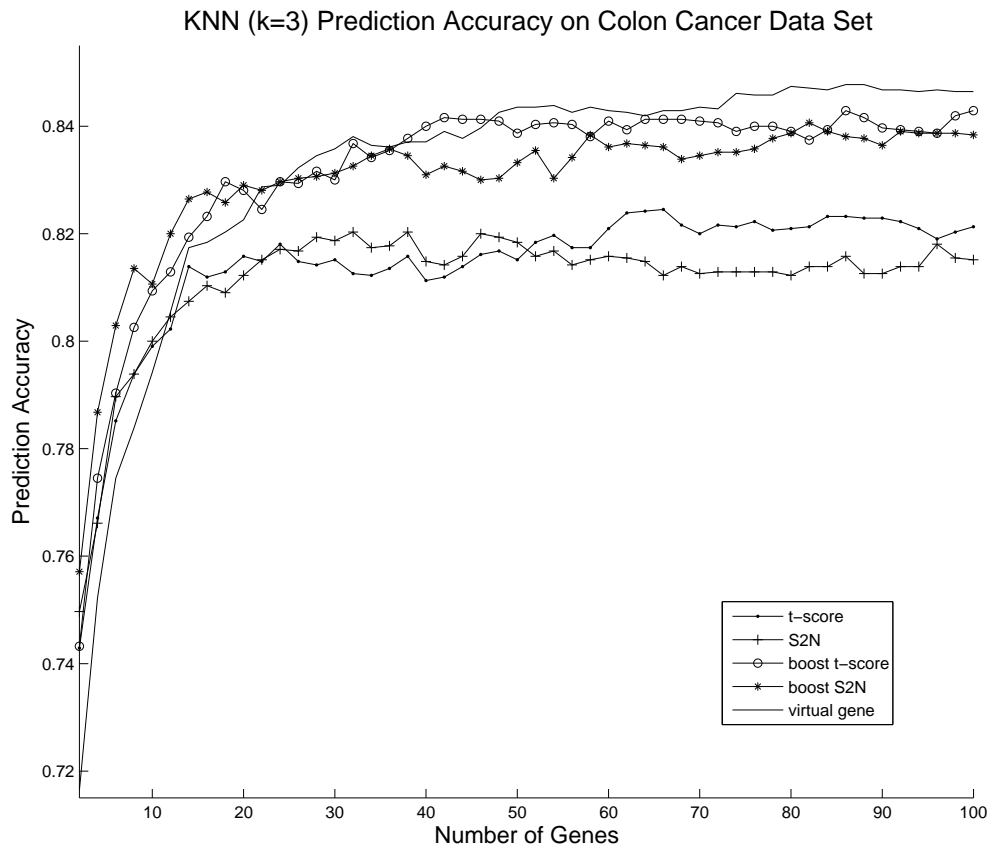


Figure 2.3.2: KNN (k=3) prediction accuracy on colon cancer data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.

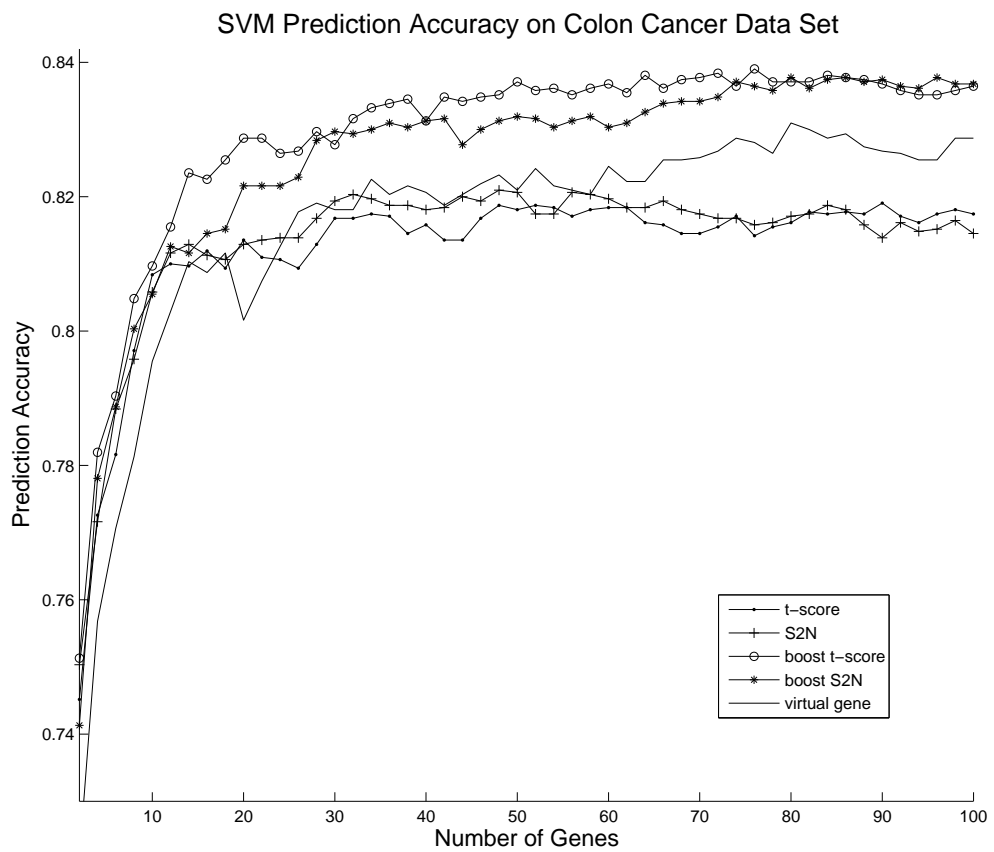


Figure 2.3.3: SVM prediction accuracy on colon cancer data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.

peaked at 4% of prediction accuracy when 100 genes are selected. In the case for KNN and SVM classifiers, boosted versions of t-score and S2N improve the prediction performance by 2-3% when 100 number of genes selected.

It is worth noting that our previously proposed virtual gene algorithm performs best when DLD classifier is used. This might be attributed to the fact that the virtual gene algorithm uses FLD (Fisher Linear Discriminant) internally. Thus when DLD classifier is used, virtual gene algorithm becomes more like a feature wrapper than a feature filter, which gives it some edges when compared to other algorithms. Using KNN classifier the performance of boosted versions of t-score and S2N becomes comparable to virtual gene algorithm. In the case of SVM classifier, boosted versions of t-score and S2N outperform virtual gene by around 1% in classification accuracy.

Comparing to values listed in Table 2.3.1, it is clear that feature selection algorithms improve classification performance on Alon data set by a big margin. The DLD classifier is most susceptible to excessive features. Using original feature space, the DLD classifier performs just a little bit better than random guessing. SVM and KNN are both more robust with excessive features. However, by using top 100 genes, we see a more than 10% hike in classification performance.

As a conclusion, BFSS algorithm performs consistently and considerably better using both t-score and S2N algorithms on Alon data set when more than 4 genes are selected.

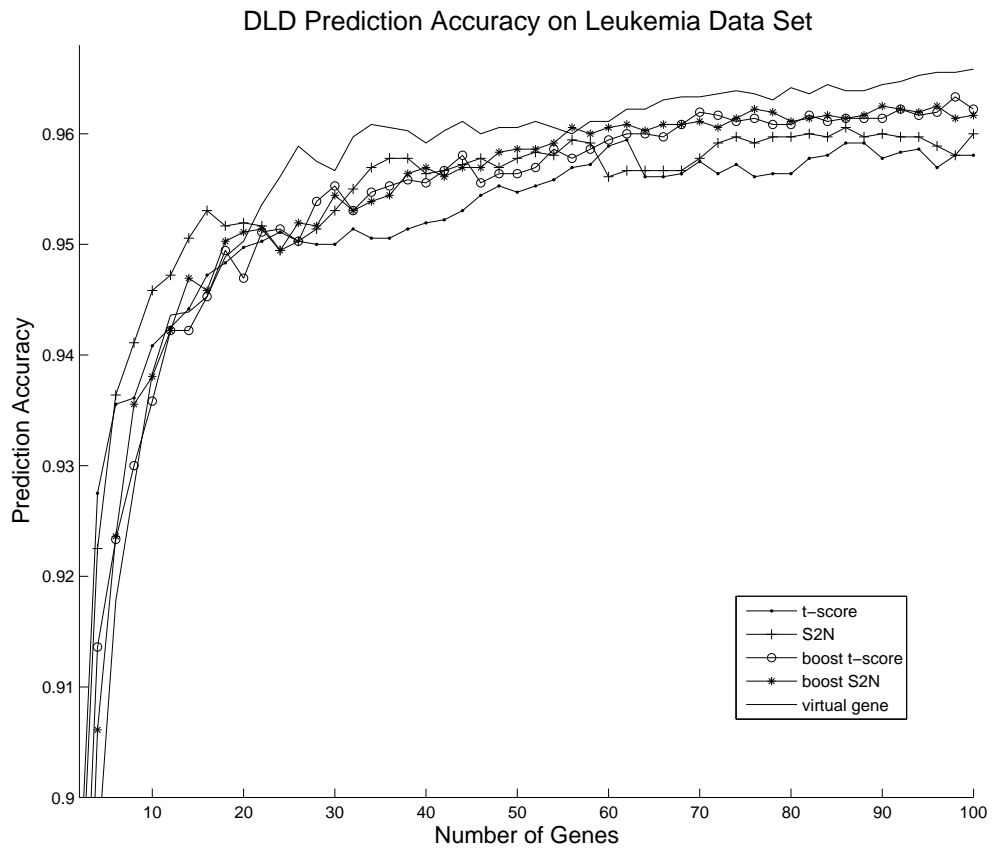


Figure 2.3.4: DLD prediction accuracy on leukemia data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.

2.3.4 Experiments on Leukemia Data Set

Experimental results on Leukemia data set are summarized in Figures 2.3.4-2.3.6. Boosted versions of t-score and S2N achieve consistently better classification accuracy than their original counterparts using all three classifiers (DLD, KNN, SVM), although in this case the improvement itself is less than what we have witnessed in colon cancer data set. This is

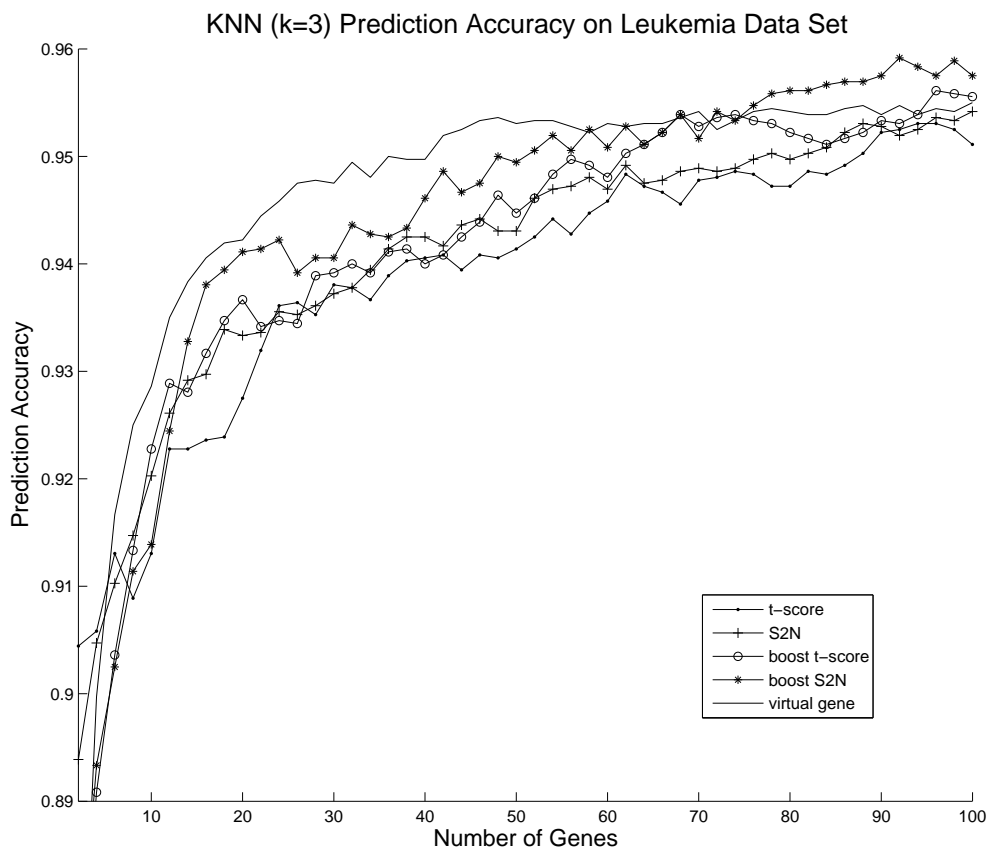


Figure 2.3.5: KNN (k=3) prediction accuracy on leukemia data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.

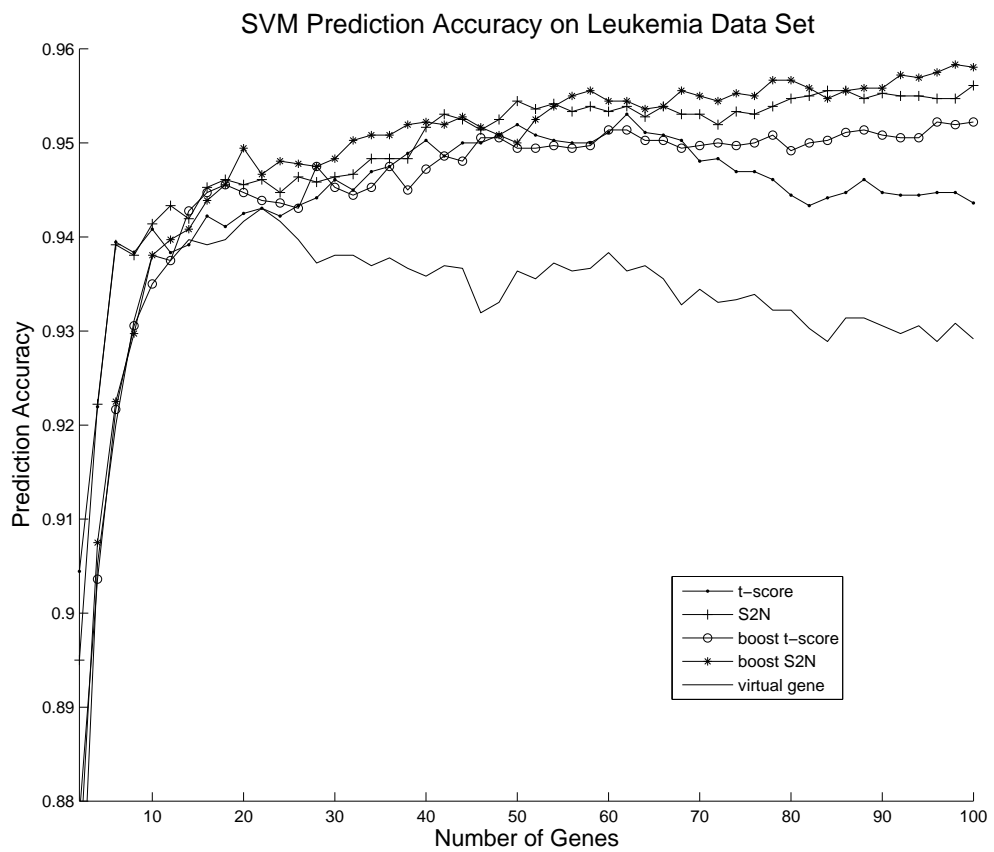


Figure 2.3.6: SVM prediction accuracy on leukemia data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.

probably due to the fact that classification accuracy on Leukemia data set is already high, there is thus less space for improvement. When using the DLD classifier, the performance increase is not much, only around 0.3%. However this gain in performance is consistent no matter how many genes are selected. When the KNN classifier is used, boosted version of S2N score performs best and outperforms the original S2N score by more than 0.5%. In the case of SVM classifier, the boosted version of t-score outperforms original t-score by as much as 1%.

When compared to virtual gene, boosted versions of t-score and S2N perform very well. Virtual gene only has some edges when the DLD classifier is used. In other cases, boosted versions of t-score and S2N perform better.

2.3.5 Experiments on Multi-class Cancer Data Set

Figures 2.3.7-2.3.9 summarize feature subset selection performance on multi-class data set.

The size of this data set is significantly larger than previously two data sets.

When the DLD classifier is used, boosted versions of t-score and S2N perform consistently better than t-score and S2N score no matter how many genes are selected as shown in Figure 2.3.7. The boosted version also outperform the virtual gene algorithm in most cases, only loosing edge to virtual gene when more than 90 genes are selected. For the KNN

classifier, both boosted versions and original versions performs considerably well, better than the virtual gene algorithm. When SVM classifier is used, both boosted versions of t-score and S2N perform better than t-score and S2N. The advantage enjoyed by boosted version is around 0.5% in terms of classification accuracy.

Comparing to Table 2.3.1, one interesting observation is that the performance of the KNN classifier is actually better using original 11985 genes than using 100 top ranked genes. In the case of SVM classifier the performance is about the same. This indicates that 100 genes may not be enough for this data set. After all only less than 1% of the features are selected. By selecting more genes we would expect the performance to further improve. We have observed the same effect in the other experiments: when the number of selected genes is too small, the classification performance suffers.

2.4 Conclusion and Discussion

In this chapter, we presented a novel general feature subset selection framework to improve the performance of single-feature based discriminative scores. In our approach, genes are selected from bootstraps of training set instead of training set itself. The sampling probability is dynamically adapted based on the performance of previously selected genes on

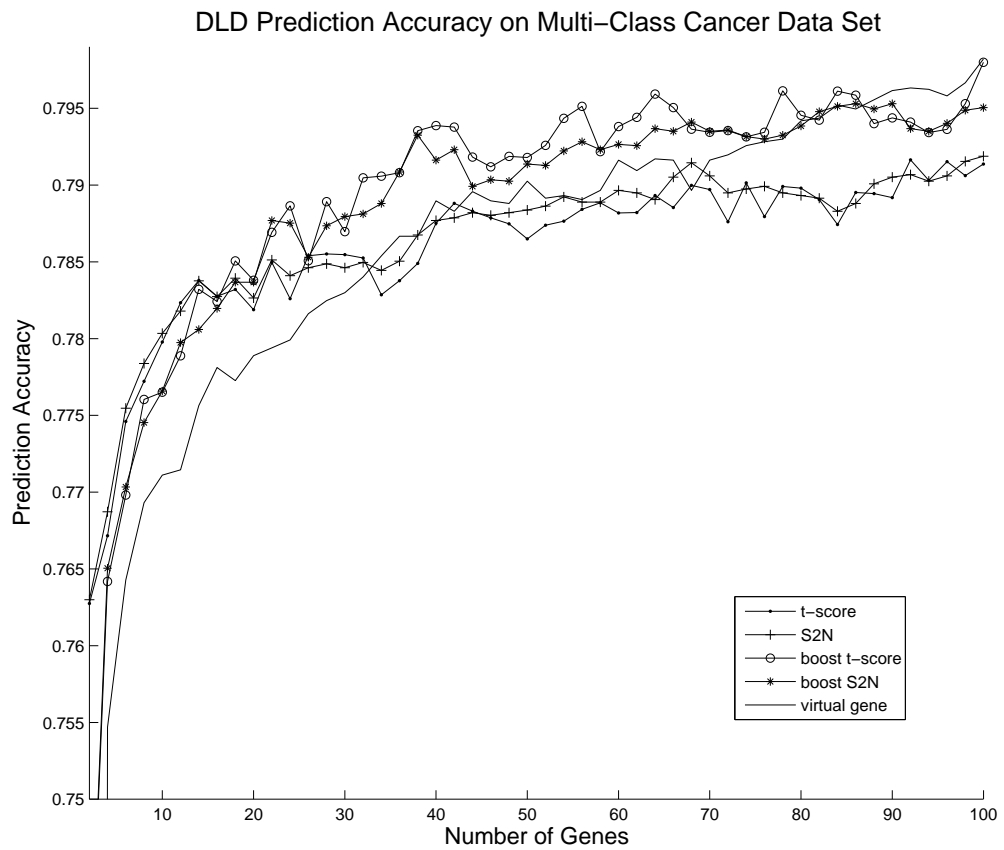


Figure 2.3.7: DLD prediction accuracy on multi-class data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.

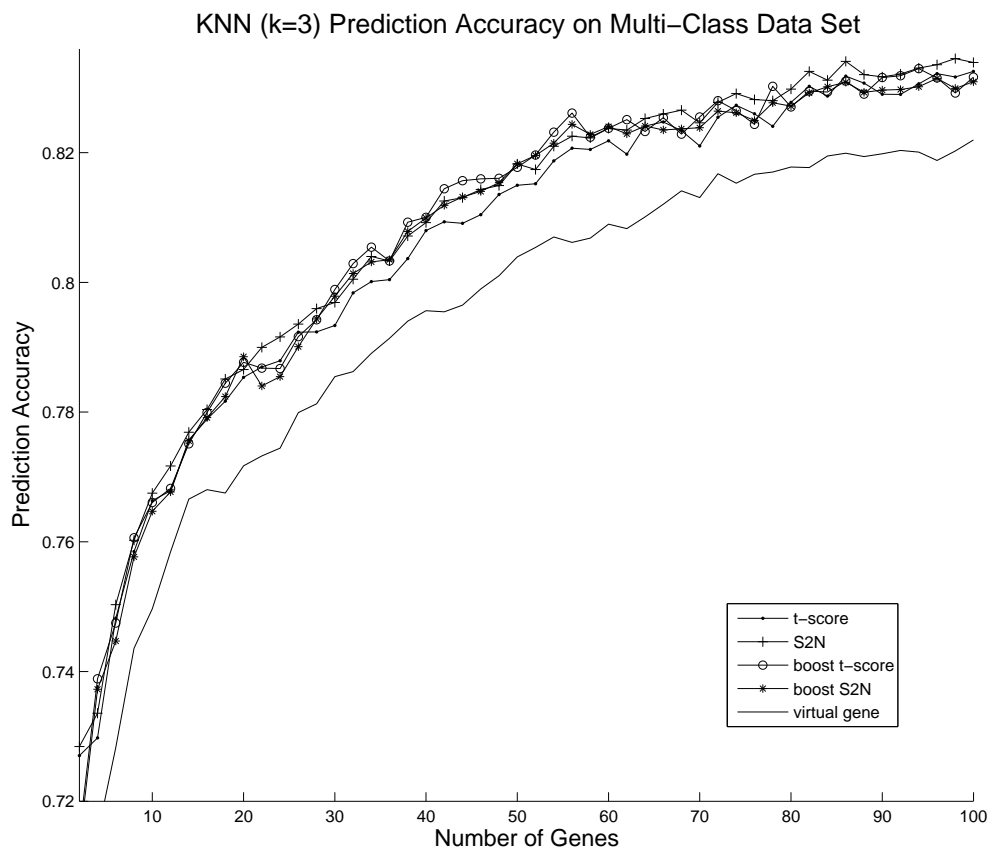


Figure 2.3.8: KNN (k=3) prediction accuracy on multi-class data set for five different feature selection algorithms: t-score, S2N, boost t-score, boost S2N and pairwise virtual gene.

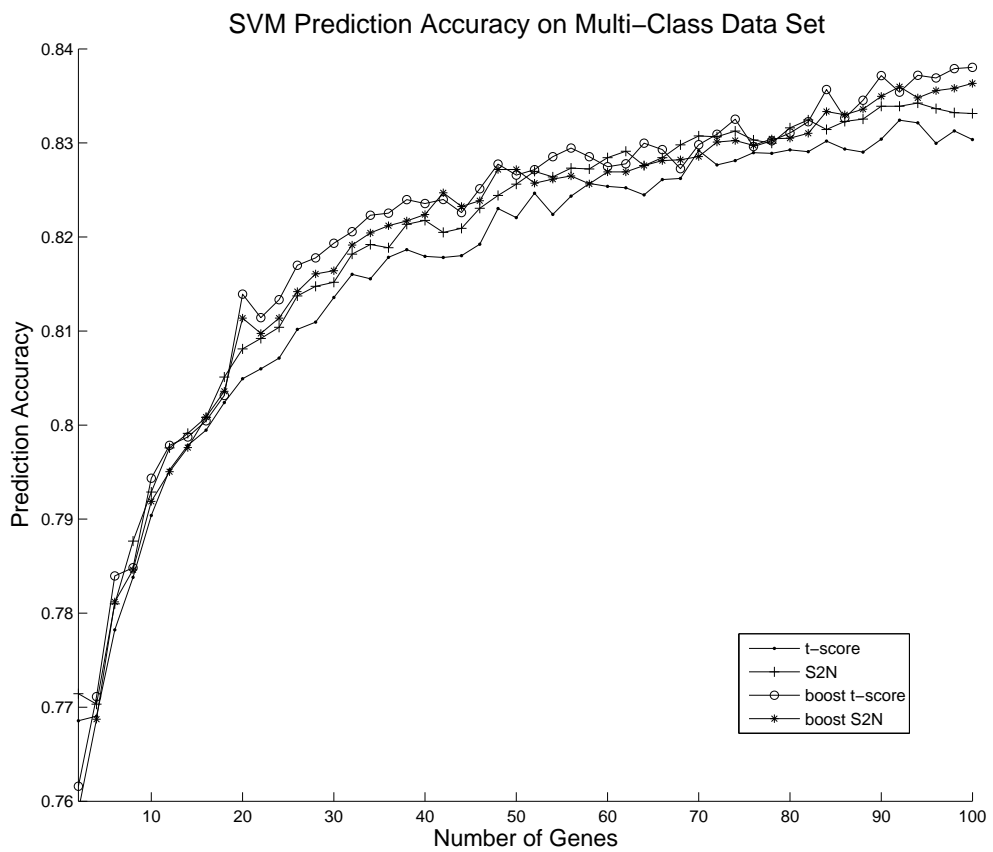


Figure 2.3.9: SVM prediction accuracy on multi-class data set for four different feature selection algorithms: t-score, S2N, boost t-score and boost S2N.

different bootstrap samples. Extensive experiments were performed on three publicly available microarray expression data sets. According to our experiments, BFSS is able to boost the performance of single-gene based discriminative scores in most cases. Boosted versions of those single-gene based discriminative scores perform consistently better in most cases and in many cases boosted versions perform considerably better than the original scores.

A nice feature of our approach is that most if not all single-gene based discriminative scores can be plugged into our system and the resulted BFSS feature selectors are expected to perform better than the original scores according to our experiments. Our approach is also independent of the classifier used. Our experiments are performed using three vastly different classification algorithms, all testifying the effectiveness of BFSS.

Although not targeted as feature selection algorithm for ensemble classifiers, our algorithm may work well with such classifiers nonetheless. [92] reported better performance could be achieved by diversifying the feature set selected for each member classifier. One extreme is actually to train member classifiers using disjoint set of features. Since our BFSS algorithm is based on bootstraps of the training set and sampling probability is dynamically adapted, diversity of the selected features is already built in.

BFSS bears some similarity to the boosting algorithms, such as AdaBoost [48, 49]. A boosting algorithm focuses on improving classification accuracy of a classifier. However,

we empirically showed in this chapter that applying boosting strategy to the feature selection stage is still beneficial. It is also interesting to examine the behavior of BFSS in conjunction with these ensemble classifiers, especially these bootstrap based ensemble classifiers (bagging and boosting). Furthermore, since BFSS and bootstrap based ensemble classifiers are all based on the bootstrapping concept, it is possible that they can be combined into a uniform framework. We are currently researching into these directions.

Chapter 3

Virtual Gene: Correlation Base Gene Selection

As mentioned in Chapter 1, traditional gene selection algorithms are mostly single-gene based. Such algorithms ignore completely correlations between genes, although such correlations is widely known. Genes interact with each other through various regulatory pathways and networks. In the previous chapter, we proposed a general feature selection algorithm called BFSS that takes into consideration the correlation between genes. Genes are selected in such a way that they perform collectively better than selecting genes only based on their discriminative scores. In essence, correlations between features are inexplicitly accounted for. In this chapter, we propose to use rather than ignore such correlations

for gene selection. We examine explicitly the correlation between gene pairs as a first step. Experiments performed on three public available gene expression data sets show promising results.

We propose a new concept called virtual gene in this chapter. A virtual gene is a linearly combined set of genes whose expression levels across different sample tissues are inferred from the expression levels of the same set of genes. Finding the best virtual gene is a problem of exponential complexity. In this chapter, we will confine us to pairwise virtual genes. In Chapter 5, gene ontology annotations are used to allow us to consider bigger virtual genes beyond gene pairs in a more intelligent manner.

This chapter is organized as follows. An illustrating example is given in Section 1. Our virtual gene algorithm is detailed in Section 2. Extensive experiments are performed and reported in Section 3. We conclude this chapter in section 4.

3.1 An Example: The Problem of Single Feature Based Discriminative Score

Consider the following two examples as shown in Figure 3.1.1. In each figure, the expression levels of two genes are monitored across several samples. Samples are labeled either

cancerous or normal. In both cases, the expression levels of the selected genes vary randomly across the sample classes. However, their correlation is a good predictor of class labels. *Virtual gene* expression level is obtained using the Definition 3.2.2. In the case of colon cancer [8] data set, the expression levels of H09719 are generally higher than that of L07648 in cancer tissues. In normal tissues, on the contrary, L07648 expresses consistently higher except in one sample. Such correlations could be good predictors of sample class labels. However, all feature selection algorithms listed in the previous section can not find and use such correlations. Single gene based algorithms will ignore both genes since neither of them is a good predictor of sample class labels in its own right. Correlation based algorithms will actually remove such correlations, should any of the genes have been selected.

3.2 Virtual Gene and Pairwise Virtual Gene Algorithm

Definition 3.2.1 *Virtual Gene* is a triplet $VG = (G_v, W, b)$ where $G_v \subseteq \mathcal{G}$ is a set of constituent genes, $|G_v| = n_v$, W is a matrix of size $n_v \times 1$, b is a numeric value. The expression levels of a *virtual gene* is determined using Definition 3.2.2.

Definition 3.2.2 (Virtual Gene Expression) Given a *virtual gene* $VG = (G_v, W, b)$ and gene expression matrix E , where $|G_v| = n_v$, E is an $n_v \times m_v$ expression matrix, the *virtual*

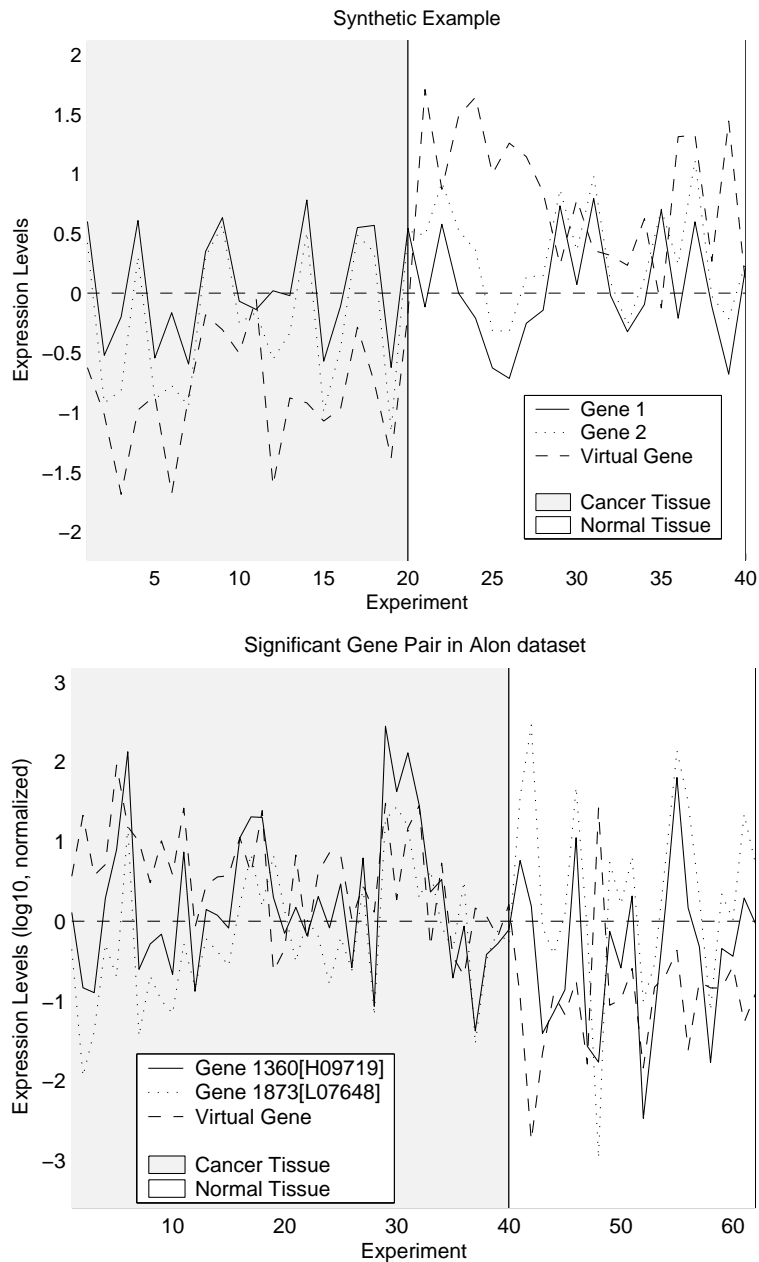


Figure 3.1.1: The idea behind the virtual gene algorithm: examples of gene pair being better predictor of class labels than any constituent single gene. Both a synthetic and a real world example are given here. The real world example comes from colon cancer data set [8]. In the figures, two genes are not good predictor of sample class labels individually. However, when combined, they become much better predictors.

gene expression VE of a virtual gene VG is a linear combination of expression matrix E .

$VE(VG, E) = W' \times E + b$, where W' is the transpose of W .

A *virtual gene* is a triplet $VG = (G, W, b)$ as defined in Definition 3.2.1. Parameters W and b are chosen using FLD (fisher linear discriminant) to maximize linear separability between sample classes as listed in Algorithm 3. Discriminative power of a *virtual gene expression* with respect to sample classes can be measured using normal single gene based scores. We use t-score in this chapter for this purpose. *Pairwise virtual gene* is a special case of *virtual gene* where the number of genes involved is limited to two. In this case, only the correlations between a pair of genes are considered. By limiting *virtual gene* to gene pairs, computation can be carried out efficiently. According to our experiments, it performs well on three public available data sets.

Algorithm 3 *gen_vg* : Calculating Virtual Gene From Training Data

Require: $\mathcal{E} = (G, S, L, E)$ as gene expression data.

Ensure: $VG = (G, W, b)$ as a virtual gene.

- 1: $(W, b) \leftarrow fld(E, L)$, (W, b) is the model returned by *fld* algorithm.
 - 2: **return** (G, W, b)
-

Definition 3.2.3 *Pairwise virtual gene and its expression are special cases for virtual gene and its expression, where the number of genes involved is limited to two.*

Exhaustive examination of all *pairwise virtual genes* requires $O(n^2)$ computation where n is the number of genes. For a large number of genes, exhaustive search of all gene

pairs becomes inefficient. Such exhaustive search also invites unwanted noise since not all gene pairs bare biological meaning. For example, for genes that are expressed in different locations in a cell, in different biological processes, without biological interactions, their relative abundance may not be biologically significant. Ideally, only gene pairs with some biological interaction shall be examined. We approximate this using a gene clustering approach. Each gene cluster corresponds roughly to some biological pathways. By limiting search among the gene pairs from the same gene cluster, we not only focus ourselves on these gene pairs that are more likely to interact biologically, but also make our gene selection algorithm much faster.

Algorithm 4 details the *pairwise virtual gene selection* algorithm. Genes are first clustered based on their expression levels. For each pair of genes in the same cluster, virtual gene expression is calculated according to Definition 3.2.2. A single gene discriminative score with respect to the sample class labels is then derived from the virtual gene expression. All within-cluster pairwise virtual gene expression scores are calculated and stored for the next stage of analysis. The best scored virtual gene is then selected and pairwise scores are modified by two parameters. Pairwise scores of virtual genes that share constituent genes with the selected virtual gene are degraded by a constant α ranging $[0, 1]$. This dampens the effect of a single dominant salient gene. In the extreme case where α is set to 0, once a virtual gene is selected all virtual genes sharing constituent genes will not be further considered. The second parameter affecting the virtual gene selection is β , which controls

Algorithm 4 *pairwise_vg* : Pairwise Virtual Gene Selection

Require: $\mathcal{E} = (G, S, L, E)$; k as the number of genes to be selected; α ; β **Ensure:** VGS: as set of pairwise virtual genes $VG = (G, W, b)$

- 1: Initialize VGS to be an empty set. Initialize *pair_score* to be a sparse $n \times n$ array.
 - 2: Cluster genes based on their expression levels in E . Result stores in *Clusters*.
 - 3: **for** each gene cluster $G' \in Clusters$ **do**
 - 4: **for all** gene $g1 \in G'$ **do**
 - 5: **for all** gene $g2 \in G'$ and $g2 \neq g1$ **do**
 - 6: $vg \leftarrow gen_vg(\mathcal{E}((g1, g2), S))$
 - 7: $ve \leftarrow VE(vg, E((g1, g2), S))$
 - 8: $pair_score[g1, g2] \leftarrow t\text{-score}(ve, L)$
 - 9: **end for**
 - 10: **end for**
 - 11: **end for**
 - 12: **for** $i = 1$ to k **do**
 - 13: $(g1, g2) \leftarrow \underset{(g1, g2)}{\operatorname{argmax}}(pair_score[g1, g2])$
 - 14: $vg \leftarrow gen_vg(\mathcal{E}((g1, g2), S))$
 - 15: add vg to VGS
 - 16: multiply *pair_score* that involves $g1$ or $g2$ by α .
 - 17: multiply *pair_score* that involves genes in same cluster of $g1$ or $g2$ by β .
 - 18: $pair_score[g1, g2] \leftarrow$ minimum value
 - 19: **end for**
 - 20: **return** VGS
-

how likely virtual genes in the same gene cluster are selected. Different gene clusters correspond to different regulatory processes in a cell. Choosing genes from different gene clusters broadens the spectrum of the selected gene set. β also ranges $[0, 1]$. In the extreme situation where $\beta = 0$, only one virtual gene will be selected for each gene cluster. After modifying pairwise scores, the algorithm begins next loop to find the highest scored virtual gene. This process repeats until k virtual genes have been selected. For performance comparison of the *pairwise virtual gene* algorithm and single gene based algorithms, each *pairwise virtual gene* counts for two genes. For example, the performance of selecting 50 genes using single gene based algorithms would be compared to performance of selecting top 25 *pairwise virtual genes*.

3.3 Complexity of the Pairwise Virtual Gene Algorithm

The *pairwise virtual gene selection* algorithm runs in three stages: (1) cluster genes based on expression profile (lines 1-2), (2) calculate discriminative scores for the *pairwise virtual gene* (lines 3-11), and (3) select pairwise virtual genes with best discriminative scores (lines 12-20). We assume gene cluster number to be θ and n, m, k, α, β as discussed above.

In the first stage of analysis, k-means algorithm runs in $O(\theta n)$. In the second stage, the actual number of gene pairs examined is $O(\frac{n^2}{\theta})$, assuming gene clusters obtained in the

previous stage are of roughly the same size. For each gene pair, the calculation of the pairwise virtual gene and its discriminative score require $O(m^2)$. Time complexity of the second stage is $O(\frac{m^2 n^2}{\theta})$. Stage three requires $O(k(\frac{n^2}{\theta} + m^2 + n + \frac{n}{\theta}))$ time. Putting them together, we have time complexity of $O(\theta n + \frac{m^2 n^2}{\theta} + k(m^2 + \frac{n^2}{\theta}))$. The most time consuming part in the previous expression is the term $O(\frac{m^2 n^2}{\theta})$. In our experiments, we choose $\theta \sim \Theta(n)$. Considering the fact that $k < n$, the time complexity of Algorithm 4 becomes $O(n^2 + nm^2)$. The $O(n^2)$ term is for k-means clustering, which runs rather quickly. If no clustering is performed in stage 1 (or $\theta = 1$, one gene cluster), the time complexity becomes $O(n^2 m^2 + kn^2)$. The savings in computation time is obvious.

Majority of space complexity for the *pairwise virtual gene selection* algorithm comes from stage 2 in the algorithm where pairwise discriminative scores are recorded. The space needed for that is $O(\frac{n^2}{\theta})$ using sparse array. Under typical situation if we choose $\theta \sim \Theta(n)$, space complexity of Algorithm 4 becomes $O(n)$, although with a large constant.

3.4 Experiments

In this section, we report extensive experimental results on three publicly available microarray data sets [8, 53, 97]. In each case, we study the gene selection problem in the context of two class sample classification.

3.4.1 Colon Cancer Data set

Data Preparation

This data set was published by Alon [8] in 1999. It contains measurements of expression levels of 2000 genes over 62 samples, 40 samples were from colon cancer patients and the other 22 samples were from normal tissue. The minimum value in this data set is 5.8163, thus no thresholding is done. We perform base 10 logarithmic transformation and then for each gene, subtract mean and divide by standard deviation. We will refer to this data set as colon cancer data set in the rest of the chapter.

Experiments

We performed three experiments on this data set to evaluate performance of the four feature selection algorithms. The main purpose of each experiment is listed as follows:

1. Compare classification accuracy and the stability of classification accuracy between *single gene t-score* [15], *single gene S2N score* [8], *clustered pairwise t-score* [15] (their all pair method modified by limiting computation within gene clusters), *pairwise virtual gene*. We refer this experiment as *alon.1* in this chapter.
2. Study how the choice of number of clusters in the *pairwise virtual gene* algorithm

affects classification accuracy and the stability of classification accuracy. We refer this experiment as *alon.2* in this chapter.

3. Study how the choice of initial cluster centers in the *pairwise virtual gene* algorithm affects gene selection performance. The *pairwise virtual gene* algorithm uses the k-means clustering algorithm to first divide genes into gene clusters. K-means algorithm is not stable in the sense that by supplying different initial cluster centers, different clustering results will be returned. We refer this experiment as *alon.3* in this chapter.

For experiment *alon.1*, we use three classification algorithms to measure the performance of the feature selection algorithms. The classification algorithms we use are KNN (k-nearest neighbor classifier, with $k = 3$) [87], SVM (support vector machine, with radial kernel) [23] and a linear discriminant method DLD (diagonal linear discriminant analysis) [83]. For cross validation of classification accuracy, we use a 2-fold cross validation method, which is the same as leave-31-out method used in [15]. We run 2-fold cross validation 100 times to obtain an estimate of classification accuracy. Standard error of classification accuracy is also reported here. The number of genes to be selected is limited to 100, as it is reported in the literature[8] that even top 50 genes produce good classifiers.

For experiment *alon.2*, we use KNN with $k = 3$ as classifier. We experimented with clustering genes into 8, 16, 32, 64, 128, 256 clusters in stage one of *pairwise virtual gene* algorithm

and then measure 2-fold classification accuracy as stated in the previous paragraph.

For experiment alon.3, we use KNN with $k = 3$ as classifier. Same experiments are repeated for 20 times with randomly generated initial cluster centers for stage one of *pairwise virtual gene* algorithm. Performance of our feature selection methods is reported.

In all experiments, we measure performance of selecting from 2 to 100 genes, increasing by 2 at a time. We set $\alpha = 0, \beta = 1$ in all these experiments. As stated before, when comparing single gene-based gene selection algorithm with *pairwise virtual gene* algorithm, we treat each *pairwise virtual gene* as two genes. Thus the performance of classifiers built from top n genes are compared with the performance of classifiers built from top $\frac{n}{2}$ *pairwise virtual genes*.

Results

Results of our experiment alon.1 are summarized in Figures 3.4.1,3.4.2,3.4.3. In each figure, the left part plots classification accuracy against number of genes used to build classifier and the right part shows standard deviations of classification accuracy. By calculating the standard deviation, we can roughly estimate how close the mean classification accuracy is to the real classification accuracy. Each figure shows classification accuracy we archived using different classification methods.

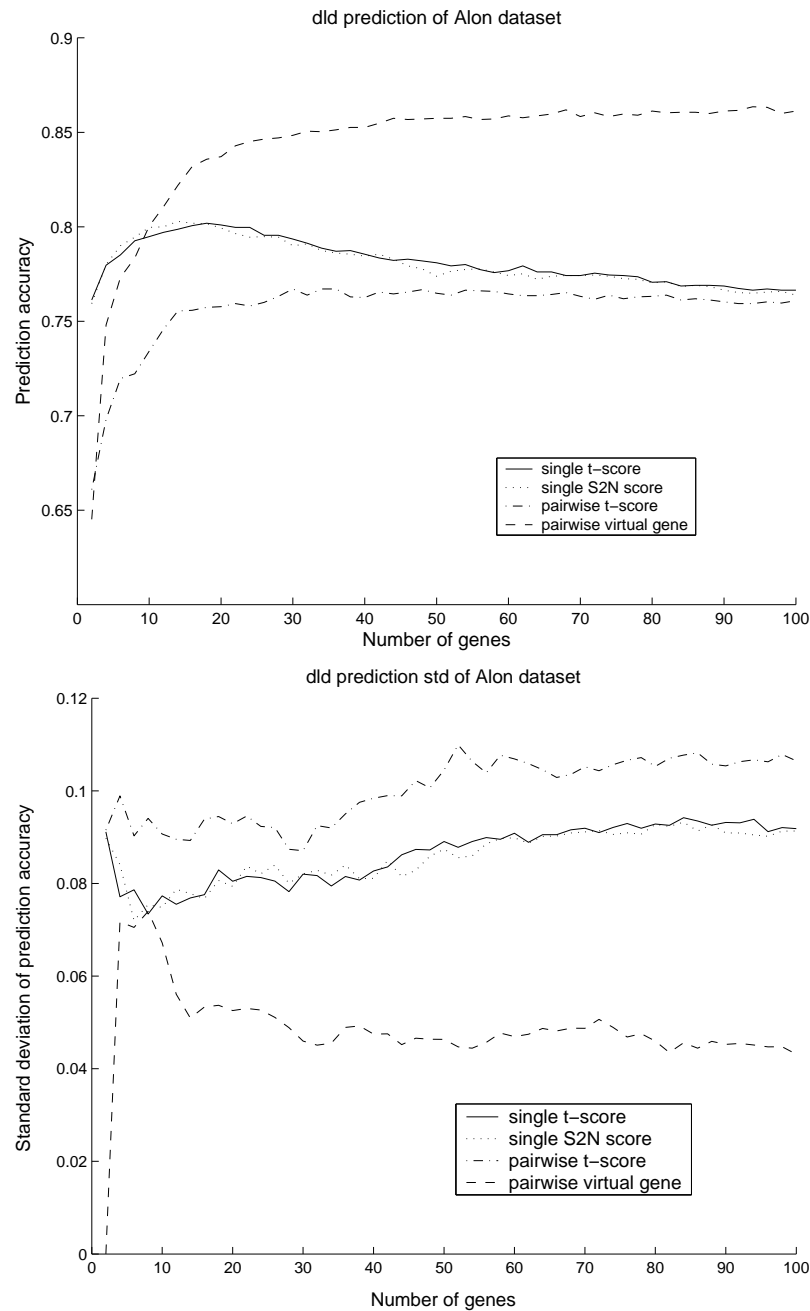


Figure 3.4.1: Result of experiment *alon.1*. Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on colon cancer data set using DLD classifier. Left figure shows prediction accuracy against the number of genes used to build DLD classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.

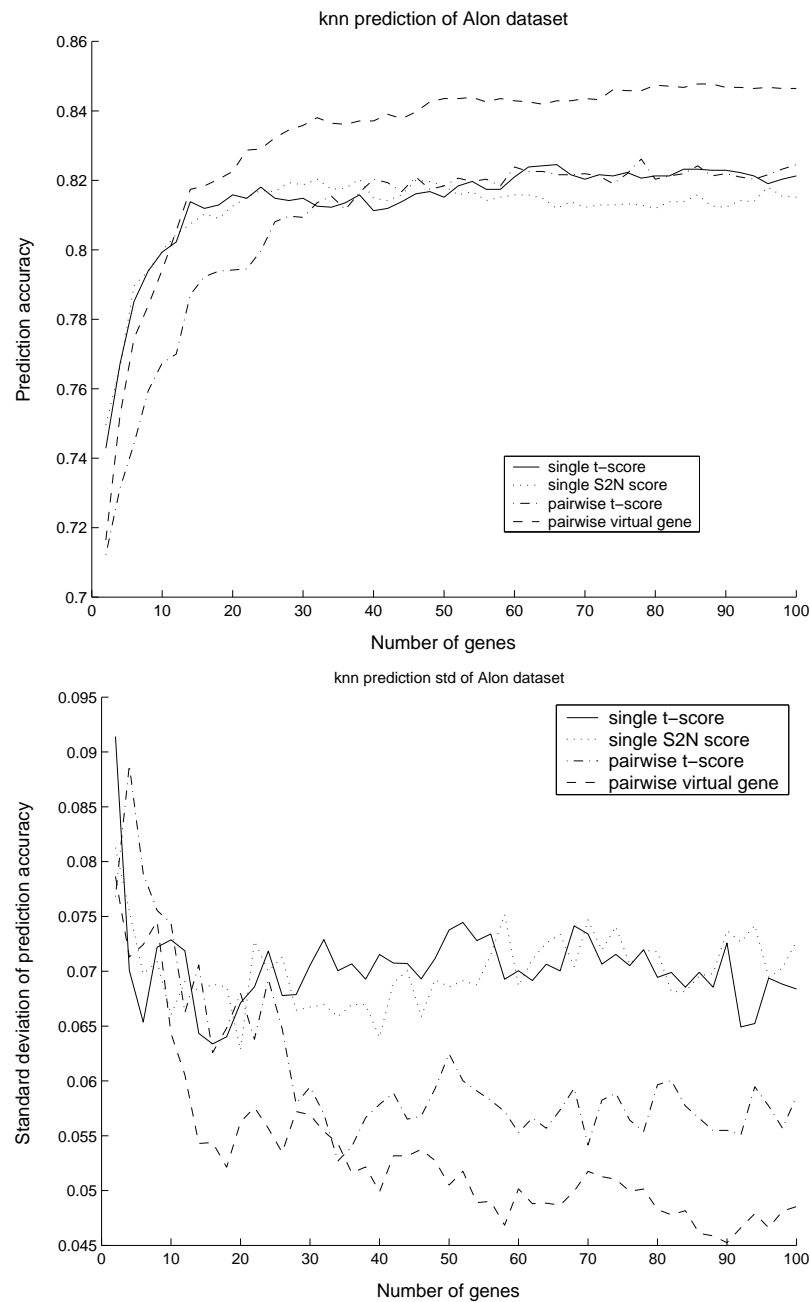


Figure 3.4.2: Result of experiment alon.2. Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on colon cancer data set using KNN classifier ($k=3$). Left figure shows prediction accuracy against the number of genes used to build KNN classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.

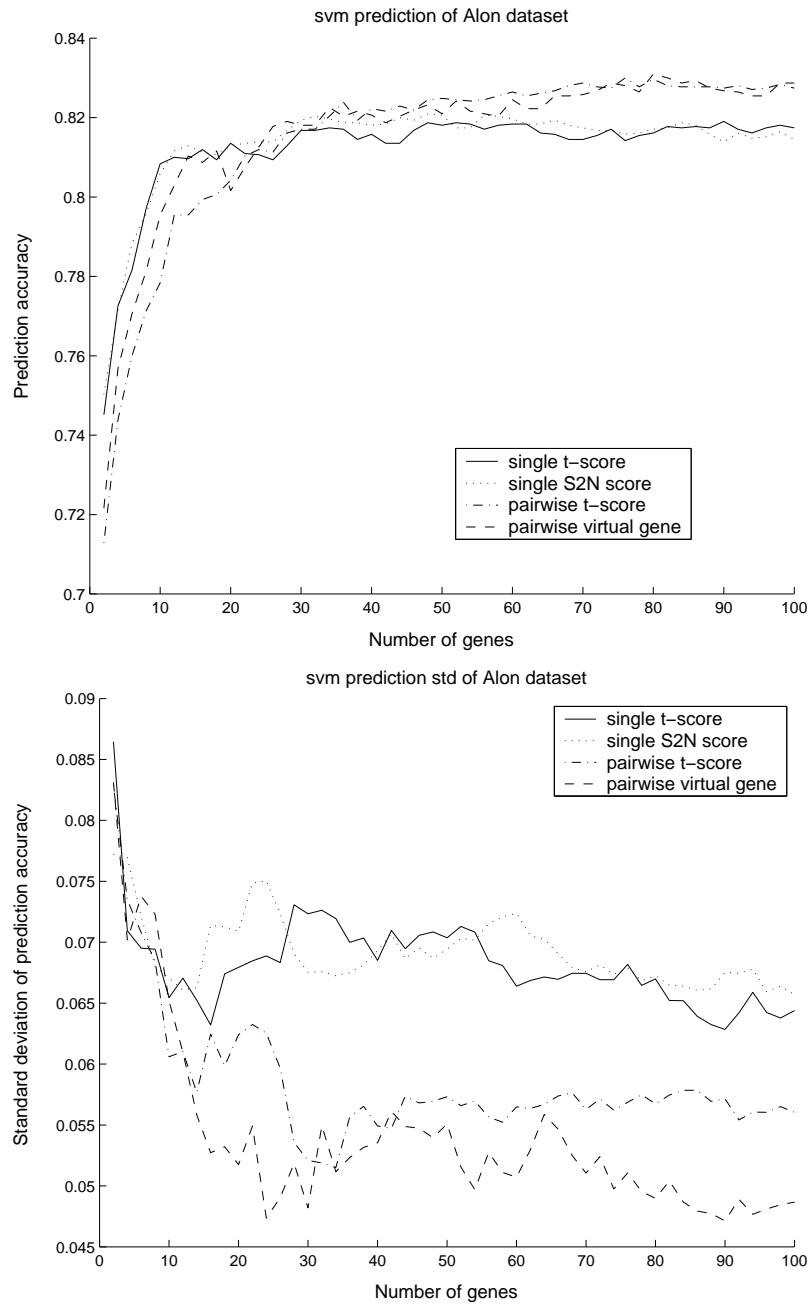


Figure 3.4.3: Result of experiment alon.3. Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on colon cancer data set using SVM classifier. In this experiment, we used a radial kernel for SVM. Left figure shows prediction accuracy against the number of genes used to build SVM classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.

From these experiments, we conclude that on colon cancer data set, the *pairwise virtual gene* algorithm performs the best. When DLD and KNN classifiers are used, *pairwise virtual gene* algorithm is significantly better than other feature selection methods we tested. When SVM is used, all FSS methods produce comparable prediction accuracy with *pairwise virtual gene* algorithm enjoying small advantage over single gene based algorithms. The *pairwise virtual gene* algorithm is also most stable, in the sense that it has the smallest standard deviation of classification accuracy.

When testing using DLD classifier, the *pairwise virtual gene* algorithm results in 5%-10% increase in prediction accuracy over other FSS methods and almost 50% decrease in its standard deviation. The experiment with KNN classifier generates similar result with the *pairwise virtual gene* algorithm leading other FSS methods in classification accuracy by 2% and having the smallest variance.

Experiments with SVM generate more mixed results in which all four FSS methods having comparable classification accuracy. The *single gene t-score* and *single gene S2N* gene selection algorithms perform better than the *pairwise virtual gene* and *pairwise t-score* algorithms when the number of genes selected is less than 20. When more genes are selected, the *pairwise virtual gene* and *pairwise t-score* algorithms perform constantly better than the *single t-score* and *single gene S2N* algorithms. When the number of genes selected is more than 50, the *pairwise virtual gene* and *pairwise t-score* algorithms outperform the other two

FSS algorithms by 1% in classification accuracy. The variations in classification accuracy still favors strongly towards pairwise methods with the *pairwise virtual gene* algorithm having the smallest variation.

For experiment alon.2, we measure the performance of the *pairwise virtual gene* algorithm setting the number of cluster in stage 1 of the algorithm to be 8,16,32,64,128,256. The results are summarized in Figure 3.4.4. We see an overall trend of decline in performance as the number of clusters increases. The classification performance peaks when 8/16 clusters are used, indicating cluster numbers suitable for this data set in that range. Compare two extremes, the 8-cluster version and the 256-cluster version, *pairwise virtual gene* algorithm performs about 2% better in classification accuracy using KNN ($k = 3$) classifier when 8 clusters are used. This is somewhat we have expected since when using 256 clusters, compared to the 8-clusters version, the computed pairwise score is around $\frac{1}{32^2}$ or around 0.1%.

It is worth noting that we used a rather crude cluster analysis algorithm, the k-means algorithm. By computing only 0.1% (or omitting 99.9%) of all possible pairs in a 8-cluster version of the algorithm, we still get strong prediction accuracy, only losing about 2% of it. This indicates that correlations between genes within clusters generated by the k-means algorithm carry much more information on sample class distinction. We also expect to further improve *pairwise virtual gene* algorithm by using more sophisticated cluster analysis

algorithms. One example is actually in Chapter 5 where domain knowledge imbedded in gene ontology annotations are used to decide which sets of genes to explore. The algorithm to find gene cliques in that chapter actually corresponds to a gene clustering algorithm. The difference is that gene clustering algorithms base their decision on gene expression profiles, yet the gene clique algorithm bases its decision in the “annotation profiles” of each gene.

Since the k-means cluster algorithm is not stable, in the sense that initial cluster center assignments will affect clustering result, we perform experiment along.3 to determine how the *pairwise virtual gene* algorithm is affected by it. We run 2-fold cross validation 100 times. Each time, the *pairwise virtual gene* algorithm is run 20 times with randomly generated initial gene clusters to select 20 different sets of virtual genes. The performance of 3-nn classifier using each of the 20 virtual gene sets is measured. Figure 3.4.5 plots the mean value of the classification accuracy, with its standard deviation. From this experiment, we conclude although k-means cluster algorithm is not stable, it performs well enough to capture important gene pairs. Twenty different initial cluster centers result in twenty different *pairwise virtual gene* selection. However, the final classification accuracy measured with 3-nn (3 nearest neighbor) classifier using these twenty different *pairwise virtual gene* selections does not vary much (having standard deviation of 0.3% to 0.5%). This justifies the use of the unstable k-means algorithm in our algorithm.

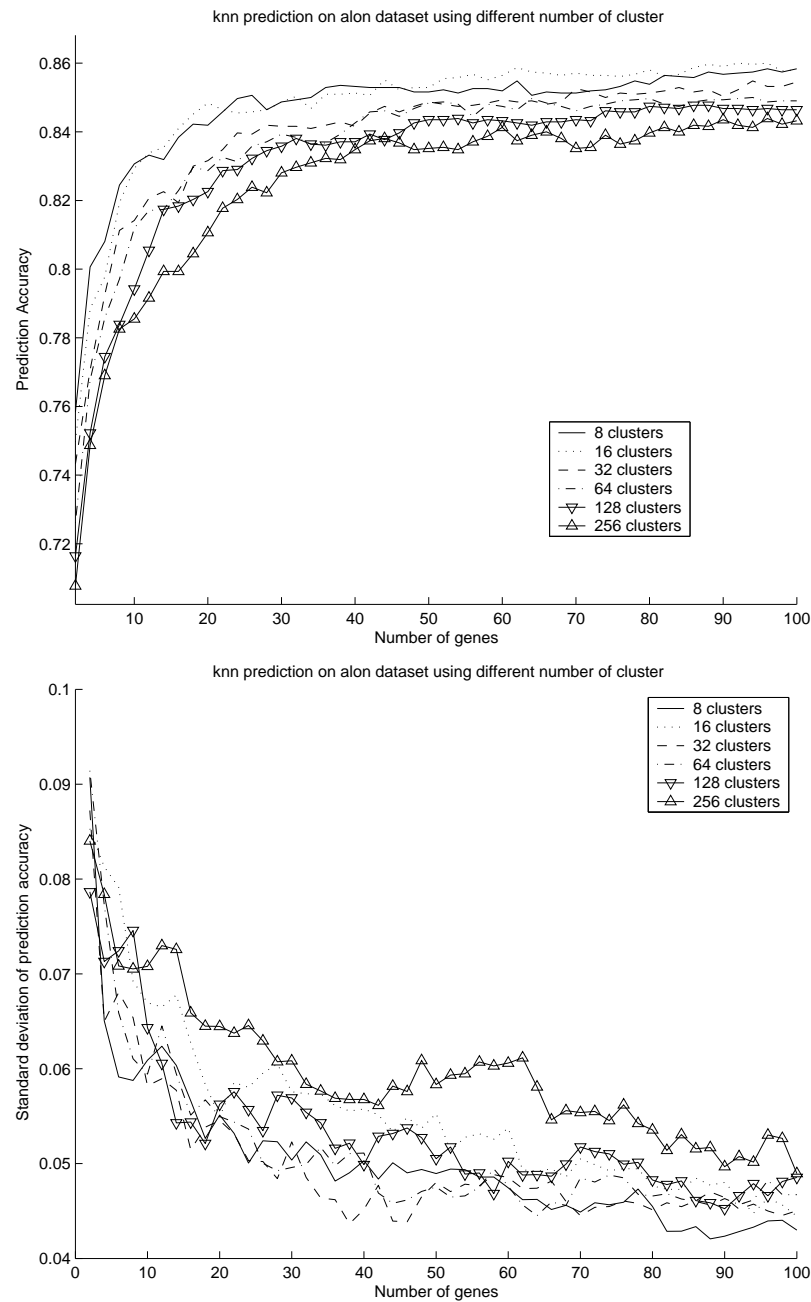


Figure 3.4.4: Prediction accuracy and its standard deviation of KNN ($k=3$) using different number of clusters in k-means algorithm (stage 1 of algorithm 4) on colon cancer data set. The prediction accuracy degrades as the number of clusters increase. However, the within-cluster gene pairs (256-cluster version vs. 8-cluster version) retain much information as a reduction of 99.9% of pairs results only around 2% decrease in prediction accuracy.

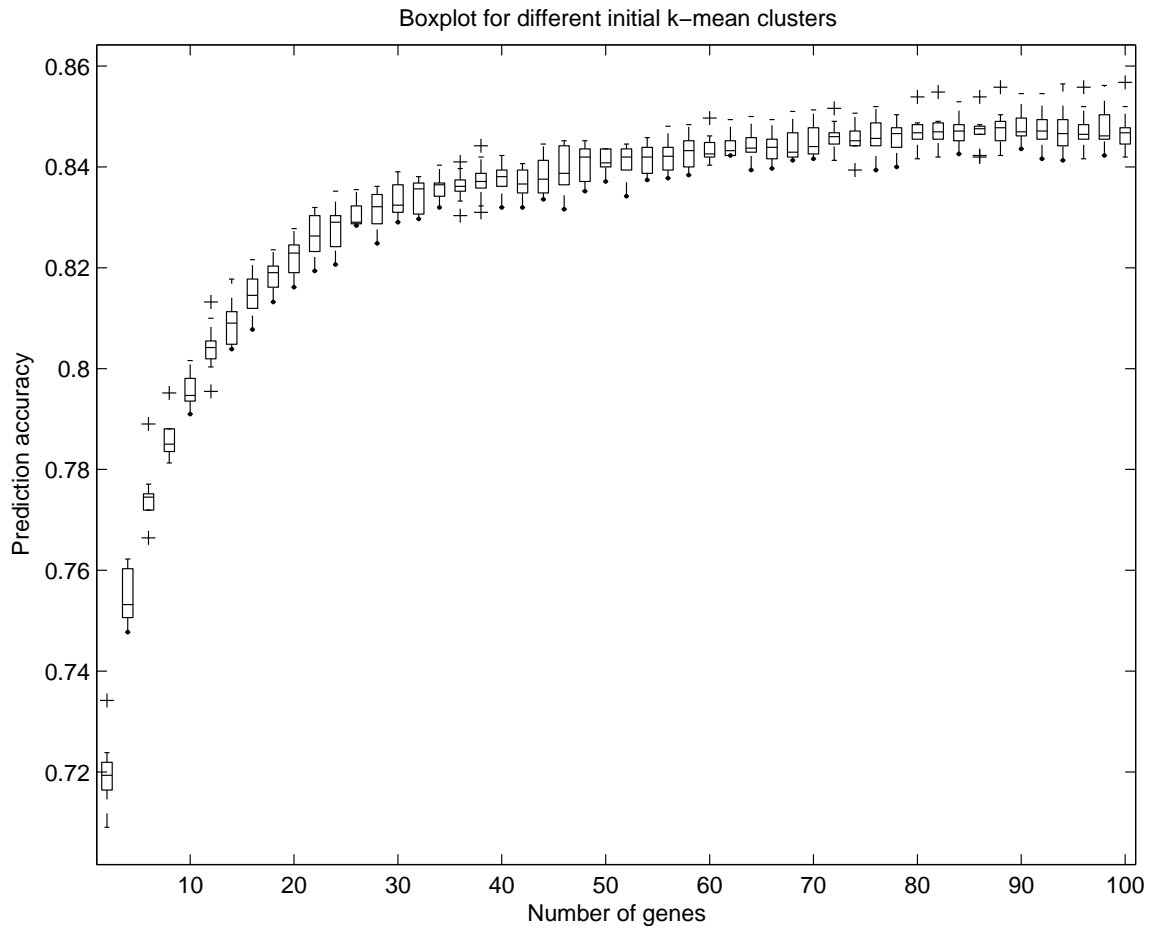


Figure 3.4.5: The boxplot of mean KNN ($k=3$) classification accuracy using *pairwise virtual gene* algorithm with 20 different initial clusters on colon cancer data set.

3.4.2 Leukemia Data Set

Data preparation

This data set was published by Golub etc. [53] in 1999. It consists of 72 samples, of which 47 samples were acute lymphoblastic leukemia (ALL) and rest 25 samples were acute myeloid leukemia (AML). 7129 genes were monitored in their study. This data set contains a lot of negative intensity values. We use the following steps (similar to Dudoit etc.[36]) preprocessing the data set before feed to our algorithm. First we threshold the data set with floor of 1 and ceiling of 16000. Then we filter out genes with $max/min \leq 5$ or $(max - min) \leq 500$, where max and min are the maximum and minimum expression values of a gene. After these two steps, the resulting 3927 genes are transformed using base 10 logarithmic and then the expression levels for each gene are normalized. We will refer to this data set as leukemia data set in the rest of this chapter.

Experiments

We perform experiments to compare feature selection performance on leukemia data set. Two classifiers (KNN, DLD) are used. Classification accuracies of these classifiers using

four feature selection algorithms (*single gene t-score*[15], *single gene S2N score*[53], *pairwise t-score*, *pairwise virtual gene*) are reported here. In all experiments, we measure performance of selecting from 2 to 100 genes, increasing by 2 at a time. We set $\alpha = 0, \beta = 0.8$ in all experiments.

Results

This data set contains roughly four times the number of genes of colon cancer data set. Straightforward computing of all gene pairs becomes intractable. Based on results obtained in the previous section on colon cancer data set, we set the number of clusters to be 256. Results are shown in Figures 3.4.6,3.4.7. For DLD classifier, when the number of selected genes is larger than 20, *pairwise virtual gene* algorithm performs consistently better than single gene based algorithms, though not by a large margin. For KNN classifier, *pairwise virtual gene* algorithm performs consistently better than all other methods we tested. Standard deviations of the classification accuracy declines as number of genes increase with one abnormal jump for single gene based methods using DLD classifier. All feature selection methods have similar variations in the classification accuracy. Overall, the *pairwise virtual gene* algorithm performs better than the single gene based algorithms on this data set.

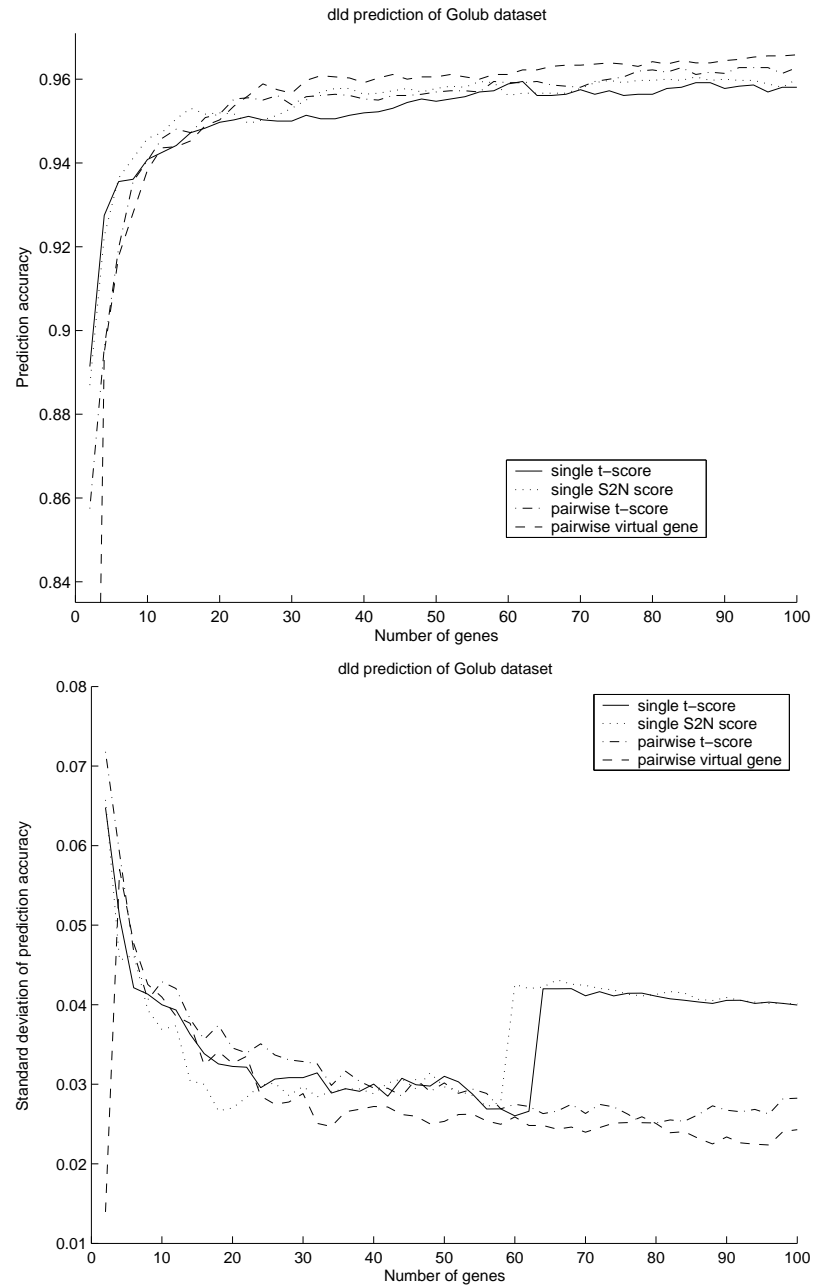


Figure 3.4.6: Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on leukemia data set using DLD classifier. Left figure shows prediction accuracy against the number of genes used to build DLD classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.

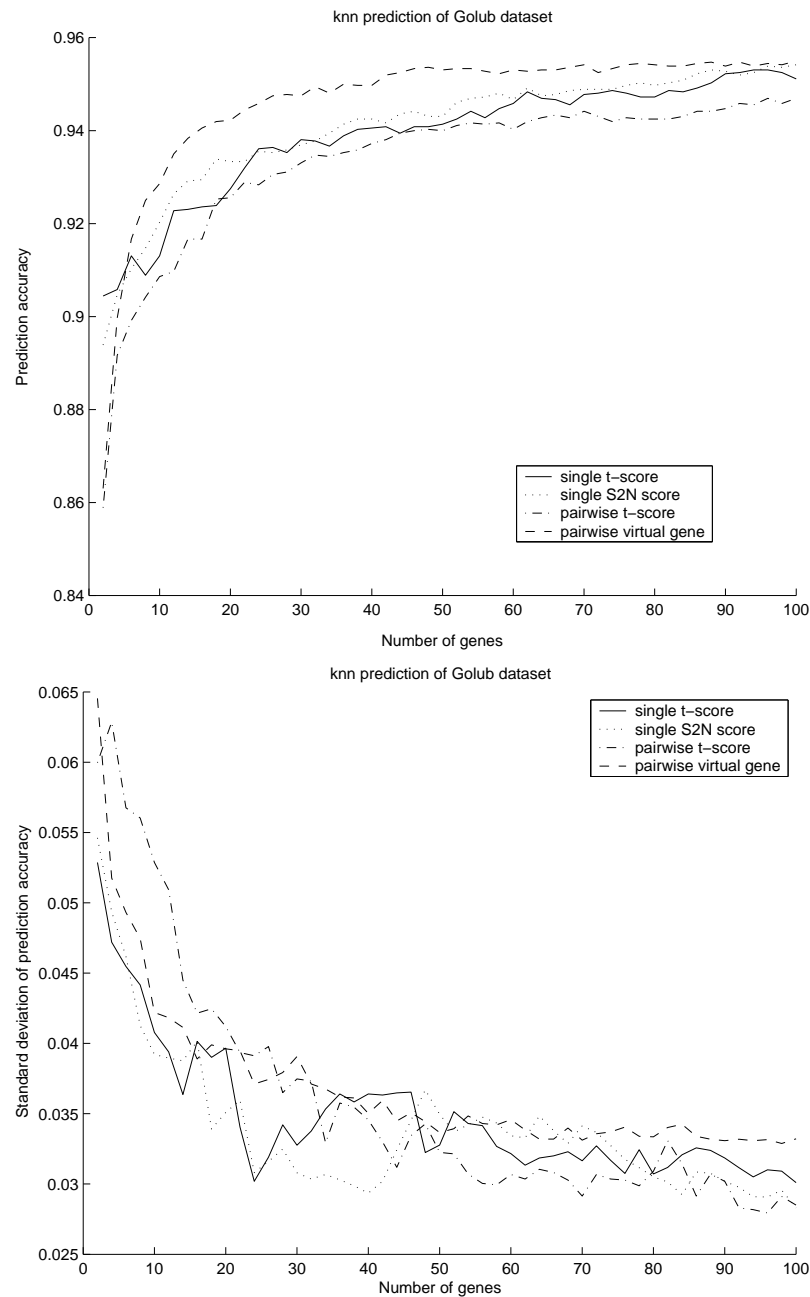


Figure 3.4.7: Prediction accuracy of four feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on leukemia data set using KNN classifier ($k=3$). Left figure shows prediction accuracy against the number of genes used to build KNN classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.

3.4.3 Multi-class Cancer Data Set

Ramaswamy etc. [97] reported study of oligonucleotide microarray gene expression involving 218 tumor samples spanning 14 common tumor types and 90 normal tissue samples. The expression levels of 16063 genes and expressed sequence tags were monitored in their experiments. The author separated the tumor samples into training set (144 samples) and testing set (54 samples). The rest 20 samples are poorly differentiated adenocarcinomas, which we did not include in our study. The training tumor set of 144 samples and 90 normal tissue samples are combined together for our study. We refer this data set as multi-class data set in the rest of our chapter.

Data preparation

Like the Leukemia data set, multi-class data set contains a lot of negative values. As a data preprocessing step, we apply a thresholding of 1 and filter out genes with $max/min \leq 5$ or $(max - min) \leq 500$. The resulting data set has 11985 genes. Logarithmic transformation and normalization are then performed before data are fed to gene selection algorithms. It is worth nothing that in the original paper by Ramaswamy, etc.[97] all 16063 genes (or ESTs) were used for classification. For our study of feature selection, the application of $max/min \leq 5$ or $(max - min) \leq 500$ filter makes sense since we are only interested in several top ranked genes.

Experiments

We measure performance of four feature selection algorithms using KNN and DLD classifiers. 2-fold cross validation is performed 100 times. Experiments are in the same setting as for colon cancer and leukemia data sets. In all experiments, we measure performance of selecting from 2 to 100 genes, increasing by 2 at a time. We set $\alpha = 0, \beta = 0.8$ in all experiments.

Results

In this experiment, we set the number of clusters to be used to 400. Result of KNN classifier shows single gene based algorithms performs better, but within 1% of accuracy compared to *pairwise virtual gene* algorithm. *Clustered pairwise t-score* algorithm performs as good as single gene based algorithms. As the number of genes selected increases, the differences in performance gradually converge.

3.5 Conclusion and Future Work

Gene selection is crucial both for building a good sample classifier and for selecting smaller gene set for further biological investigation. Feature extraction algorithms (PCA, SVD,

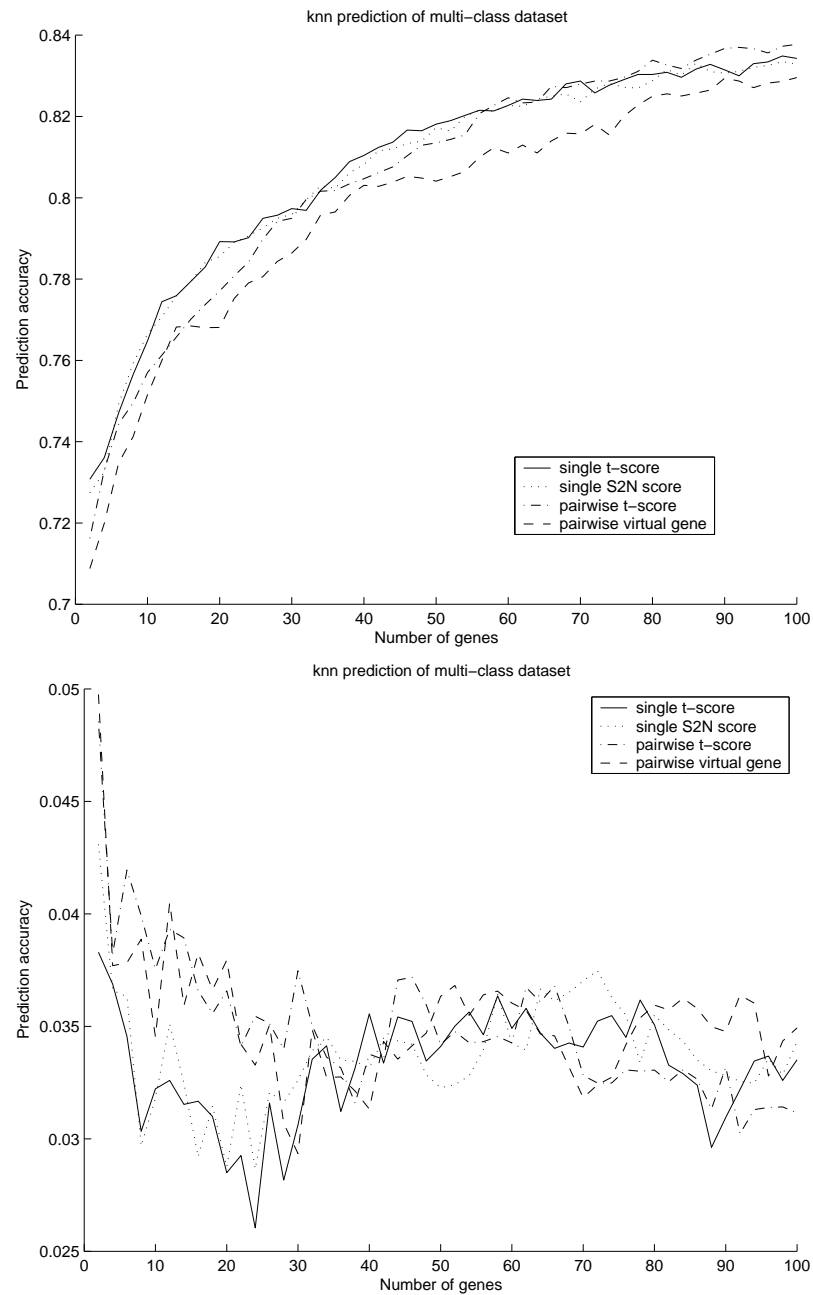


Figure 3.4.8: Prediction accuracy of 4 feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on multi-class data set using KNN classifier ($k=3$). Left figure shows prediction accuracy against the number of genes used to build KNN classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.

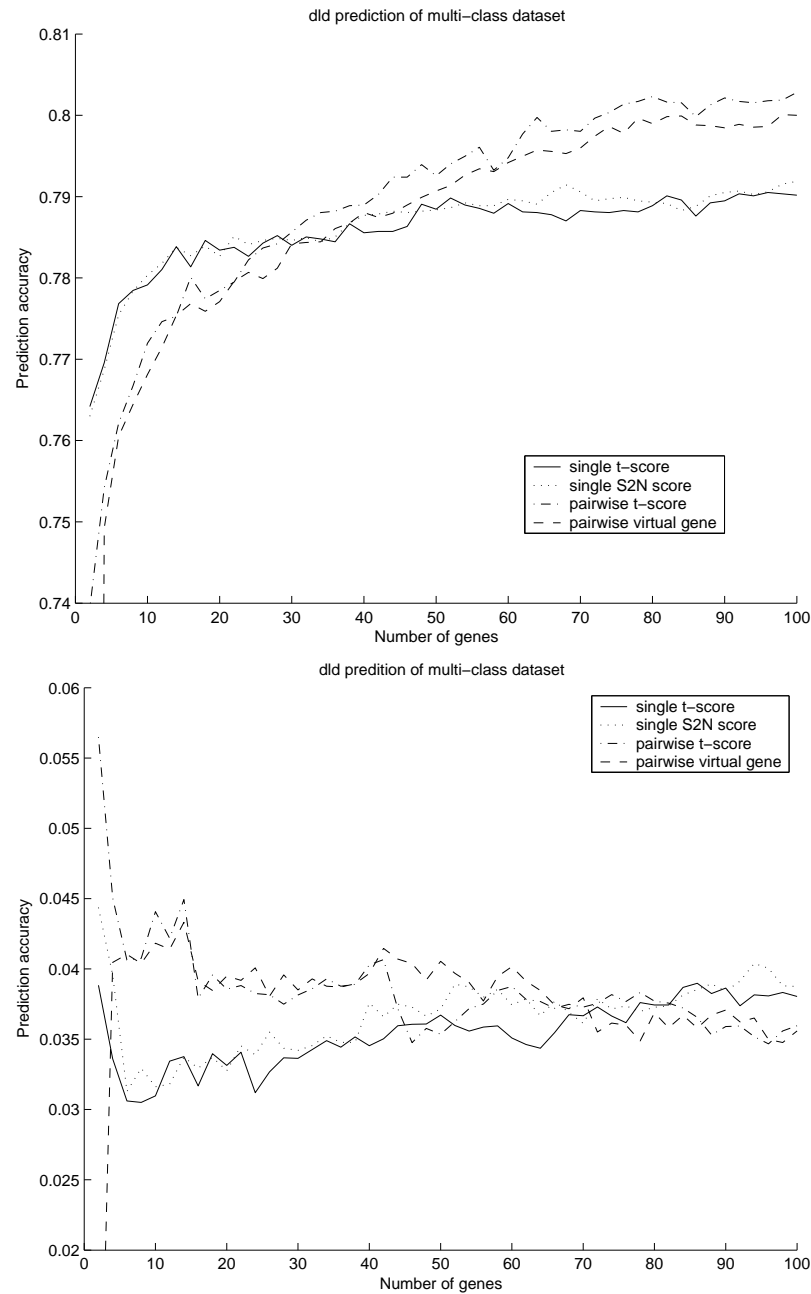


Figure 3.4.9: Prediction accuracy of 4 feature selection methods: t-score, S2N, pairwise t-score and pairwise virtual gene on multi-class data set using DLD classifier. Left figure shows prediction accuracy against the number of genes used to build KNN classifier. Right figure shows the standard deviation of prediction accuracy against the number of genes.

etc.), single gene based discriminative scores (t-score, S2N, TNoM, information gain, etc.) and correlation based algorithms have been proposed for this purpose. In this chapter, we proposed a totally different approach. Instead of trying to minimize correlations within the selected gene set, we examined whether such correlations are good predictors of sample class labels. *Virtual gene* is a linear combination of a set of real genes. Our experiments confirm our assumption that the correlations between genes are indeed good predictors of sample class labels, better in many cases than single gene based discriminative scores. There are biological explanation for this: genes interact with each other. The relative abundance of genes is a better predictor than the absolute values. Using gene clustering algorithms to limit gene pair selection seems promising. Our experiments show that by calculating pairwise scores for only a very small portion (0.5%) of all possible gene pairs, decent classification performance can be achieved. This in turn shows most useful pairwise correlations are contained within gene clusters.

Our algorithm still has space for improvement. First but not least, we are interested in combining single gene based scores and virtual gene. In contrast to correlation based gene selection approaches, we can select top genes with high individual scores and top correlations between genes. We used gene clustering algorithm to identify some biologically related genes. Gene clusters represents genes with homogeneous expression profiles. We have focused on the gene correlations within gene clusters. It is also interesting whether gene correlations between gene clusters could be used for our purpose of effective sample

classification. There are actually gene selection algorithms that use such intuition. For example the Gene Shaving algorithm proposed by Hastie et al. in [59].

We also want to examine larger virtual genes, virtual genes that combine more than two genes. Gene clustering is only a crude way of grouping co-regulated genes. In Chapter 5 we will present an extension of virtual gene algorithm that takes GO annotations into consideration. Our algorithm is quite open, several other algorithms (e.g., cluster analysis and discriminative power of single gene) can be plugged into our algorithm without much modification. We leave this as future work as well.

Chapter 4

Dealing with False Positives Using Gene Ontology

In the previous two chapters, we described two novel gene selection algorithms that take into consideration relationships between genes. Improvement in sample classification accuracy is observed. However, due to the characteristics of microarray technology and the underlying biology, namely large number of genes and limited number of samples, the statistical soundness of gene selection algorithm becomes questionable [31]. One major problem is the high false discover rate. Microarray experiment is only one facet of current knowledge of the biological system under study. In this chapter, we propose to alleviate this high false discover rate problem by integrating domain knowledge into the

gene selection process. Gene Ontology represents a controlled biological vocabulary and a repository of computable biological knowledge. It is shown in the literature that gene ontology-based similarities between genes carry significant information of the functional relationships [11, 124]. Integration of such domain knowledge into gene selection algorithms enables us to remove noisy genes intelligently. We propose an add-on algorithm applied to any single gene-based discriminative scores integrating domain knowledge from gene ontology annotation. Preliminary experiments are performed on publicly available colon cancer data set [8] to demonstrate the utility of the integration of domain knowledge for the purpose of gene selection. Our experiments show interesting results.

4.1 The Problem of Gene Selection on Small Sized Samples

In this section, we give an example of the problems facing gene selection from limited samples. Alon [8] published their experimental results on colon cancer. The data set consists of 62 microarray experiments on normal and colon cancer tissues. Expression levels of 2000 genes are monitored. After log-transformation, expression levels in this data set range from 0.76 to 4.32 with mean of 2.30 and standard deviation of 0.49. This is a commonly used benchmark data set for data analysis algorithms on microarray data sets.

Table 4.1.1: Gene selection on randomized data set. Each entry shows the number of genes that score greater than cutoff t-scores in each data set (original and random generated.)

<i>Number of Genes</i>	<i>Top 50</i>	<i>Top 100</i>	<i>Top 200</i>
<i>Cutoff t-scores</i>	3.82	3.23	2.7
Alon Orig log10	50	100	200
Random uniform	1.06 ± 1.04	5.32 ± 2.28	21 ± 4.41
Random normal	0.81 ± 0.89	4.69 ± 2.19	19.55 ± 4.43
Random empirical	0.71 ± 0.83	4.475 ± 1.97	19.5 ± 4.04
Random relabel	0.86 ± 6.39	4.50 ± 22.35	20.98 ± 62.32

Let us examine the statistical significance of gene selection on such data sets. We use four approaches to assess the false discovery rate of gene selection algorithms based on t-scores by calculating t-scores for randomly generated gene expression arrays. In the first two approaches, we generate random expression arrays of the same size (2000×62) from uniform/normal distribution with same parameters as original log transformed data set. We also experimented generating from empirical distribution of original log transformed data by assuming the histogram as an approximate probability density function. Later we generate random expression array by randomly reassigning sample class labels. The cutoff t-scores for choosing top 50, 100 and 200 genes from the original log transformed data set are 3.82, 3.23 and 2.7, respectively. Experimental results summarized in Table 4.1.1 show the number of random genes having larger-than-cutoff t-scores in each case. Those random genes would have been selected by t-score based algorithms. All the random experiments are repeated 1000 times.

From these experiments we observe that even randomly generated expression levels may

result in high discriminative scores. Suppose we merge the original data set with a random data set, resulting in a 4000 genes by 62 sample expression matrix. If we were to choose top 200 genes using t-scores from such merged data set, more than 10% of selected genes would come from the random generated portion of data. The situation becomes more hopeless if we add more random genes or the number of samples is even smaller. When 8000 random genes are added to the merged data set, the probability of selecting such random genes in top 300 gene list is more than 30%.

4.2 Integrating Biological Knowledge into Gene Selection

One way to overcome this apparent drawback in the feature selection process is the integration of domain knowledge. Biologists have long been doing this. Gene selection is only the starting point in many biological studies. Genes may be added/removed to/from selected gene set at a later stage pending on other biological evidences. However, to our surprise, there are not many feature selection algorithms proposed in the literature to actually utilize domain knowledge. Although there are already a plenty of online computable biological knowledge bases.

In this section, we propose an add-on algorithm to existing single gene-based gene selection algorithms. Given a single gene-based discriminative scores (of the sample class labels),

our algorithm processes the score using biological information contained within GO annotation. This results in a new class of discriminative scores prefixed with the name "GO adjusted". This section is divided into several subsections. Background and notation used are given in the first subsection. These notations will be used throughout this chapter. We will then elaborate definitions of our algorithm. Time and space complexity of algorithm is discussed in Subsection C. In the last subsection, we layout an experimental framework for testing the effectiveness of our algorithm.

4.2.1 GO Adjusted Scores

Definition 4.2.1 *Informative Genes* are those genes having discriminative scores larger than θ , or $\mathcal{F}(g) > \theta$.

Definition 4.2.2 *Discriminative Power* of a GO term is defined as the percentage of informative genes among all genes that are annotated with such GO term.

$$DP(go) = \frac{|\{g | g \in go \wedge \mathcal{F}(g) > \theta\}|}{|go|} \quad (4.2.1)$$

Definition 4.2.3 *Informative GO Term* is defined as those GO terms go whose discriminative power is larger than γ and the number of informative genes annotated with go is larger than β .

$$DP(go) > \gamma \quad \mathbf{and} \quad |\{g | g \in go \wedge \mathcal{F}(g) > \theta\}| > \beta \quad (4.2.2)$$

Definition 4.2.4 *GO adjusted discriminative score* is defined using one single gene discriminative score $\mathcal{F}(g)$ and a GO term go , where $g \in go$.

$$\mathcal{F}_a(g, go) = \begin{cases} 0 & \text{if } go \text{ is not informative;} \\ \mathcal{F}(g) & \text{if } go \text{ is informative.} \end{cases} \quad (4.2.3)$$

Definition 4.2.5 *Best GO adjusted discriminative score* is defined as the best GO adjusted discriminative score out of all possible GO annotations of a single gene.

$$\mathcal{F}_b(g) = \max_{\forall go, g \in go} \mathcal{F}_a(g, go) \quad (4.2.4)$$

In this section, we present our gene selection algorithm. Given a single gene-based discriminative score \mathcal{F} and three parameters θ, β, γ , our algorithm calculates a modified single gene-based discriminative score named “best GO adjusted score.” The basic idea behind our algorithm is straightforward. While the expression levels of random genes may correlate with sample class labels by chance, it is far less likely that majority of these random genes will also have common GO annotation. In other words, it is far less likely that those random genes will have valid biological connections, either participating in the same biological processes, or manifesting the same molecular functions, or being found in the same cellular components.

Informative genes are defined by Def. 4.2.1 to be those genes whose single gene discriminative scores \mathcal{F} pass threshold θ . Discriminative power of a GO term is defined by Def. 4.2.2 as the percentage of informative genes among all genes that are annotated with the GO

term in question. Discriminative power of a GO term with respect to sample class labels measures the collective discriminative power of genes annotated with that GO term. This in turn measures how different biological processes, cell components and molecular functions are affected under different experimental conditions. The higher discriminative power of a GO term, the stronger a GO term is correlated with sample class labels. The value of θ has same range as the corresponding single gene discriminative score, which is the user's estimate of what a significant score is for \mathcal{F} . We further call a GO term informative GO term if such a GO term satisfies the two conditions in Def. 4.2.3. First, more than β informative genes needs to be annotated by a GO term in order for that GO term to be called informative. Secondly, the percentage of informative genes among all genes annotated by the GO term needs to surpass threshold γ . These two criteria are set to fend off the effect of random genes. We will discuss how to choose these parameters later.

Single gene-based discriminative score \mathcal{F} is then modified according to the discriminative power of GO terms. To get rid of noisy genes, informative genes are only selected from those informative GO terms. We essentially strengthen single gene-based scores if significant amount of other genes that share common known biological annotation with the given gene are also discriminative of sample class labels. We define "GO adjusted discriminative score" $\mathcal{F}_a(g, go)$ according to Def. 4.2.4. The score is 0 if the annotating GO term is non-informative, otherwise it is the same as the single gene discriminative score, or $\mathcal{F}_a(g, go) = \mathcal{F}(g)$. Here we assume single gene-based discriminative score is positive and

the larger the score is, the more discriminative the corresponding gene is. Each gene product is annotated with potentially multiple GO terms. We define “best GO adjusted score” $\mathcal{F}_b(g)$ to be the best “GO adjusted score” for a gene among all its annotation. We assume the transitivity of gene annotation in this work. If the direct annotating GO term of a gene is not informative, the gene may still be considered if any parent GO terms of the direct annotating GO term are informative.

The algorithm to calculate “best GO adjusted score” is divided into three distinct stages. In the first stage, single gene-based discriminative scores are calculated using traditional methods. In the second stage, we compute for each GO term its discriminative power using Algorithm 5. In the third stage, single gene-based discriminative scores and discriminative power of GO terms are combined to produce the final score using Algorithm 6. We will elaborate computation in stages two and three.

In order to compute stages two and three efficiently we define several data structures as illustrated in Fig. 4.2.1. The central data structure is a DAG of GO terms. Each node in this DAG contains the following information: the accession number of this GO term, the genes that are annotated with this GO term, the informative genes (those genes whose $\mathcal{F}(g)$ pass threshold θ) that are annotated with this GO term, list of genes that this GO term directly annotates, list of children GO term nodes in the DAG and a flag “visited” for DAG transversal algorithms. Three lists are also defined to facilitate computation. First is the list

of single gene-based discriminative scores for each gene. The second list is the list for each gene pointers to its GO terms. The third list is a list of GO terms in the DAG.

We add a pseudo GO term “root” which is the common parent of the three ontology branches. In this way, gene ontology becomes single rooted. We do notice biologists in different domain may want to use different portion of gene ontology. No matter what portion of GO is of interest, our algorithm remains the same and needs only minor modifications. Algorithm 5 employs a depth first traversal of the GO DAG from this pseudo root node. In each visitation of a node, the lists of genes/informative genes that this GO term annotates are calculated from the lists of genes/informative genes that this GO term directly annotates and the lists of genes/informative genes its children GO terms annotate. Later Algorithm 6 uses the genes/informative genes lists for each GO term to derive both discriminative power of a GO term and the best GO adjusted scores for each gene.

4.2.2 Complexity of Best GO Adjusted Score

We analyze the computational cost for the best GO adjusted score. Assume the number of GO terms to be $|GO|$, the number of genes to be $|G|$, the average number of genes to be annotated by a GO term to be μ . Assume the average branching factor of the GO DAG (only involving GO terms) to be ξ .

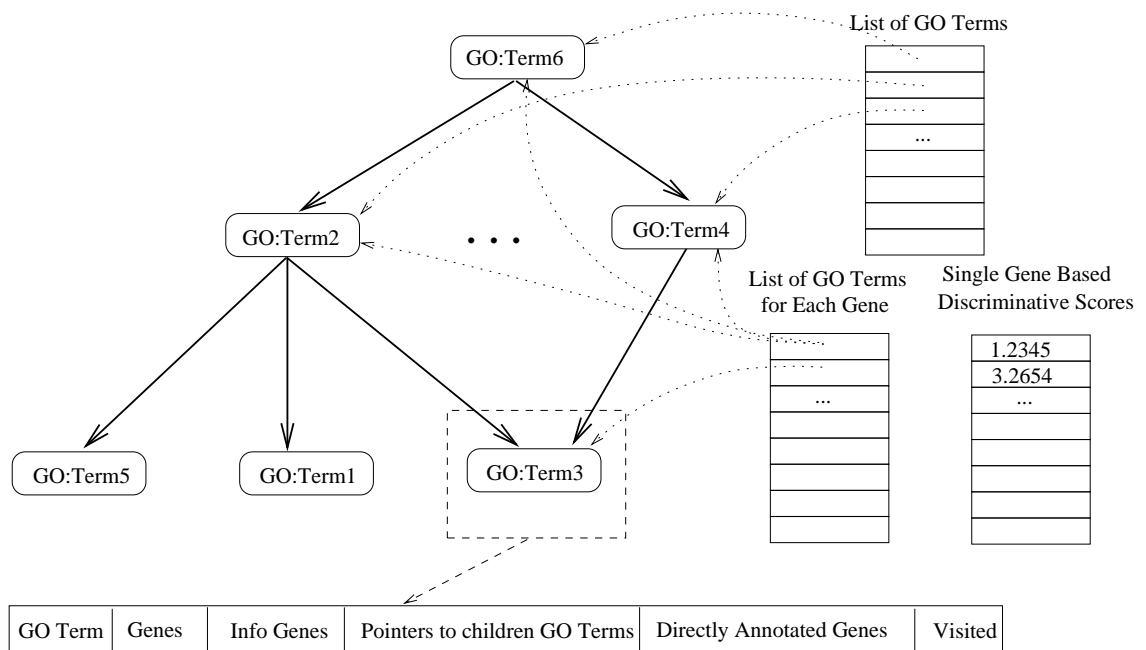


Figure 4.2.1: Main data structures used in Algorithms 5 (*GO_gene_counter*) and Algorithm 6 (Best GO Adjusted Score).

Algorithm 5 *GO_gene_counter*($go_{root}, scores, \theta, \in, \leftarrow$) Compute genes/informative genes for each GO term

Require: go_{root} : root of the GO DAG in Fig. 4.2.1, we use \in, \rightarrow as described before; $scores$ as an associative array such that $scores(g) = \mathcal{F}(g)$; θ as threshold score.

Ensure: The *Genes* and *Info_genes* for each GO node properly set

```

1: if not  $go_{root}.visited$  then
2:   for all  $gene \in go_{root}$  do
3:     Add  $gene$  to  $go_{root}.Genes$ 
4:     if  $scores(gene) > \theta$  then
5:       Add  $gene$  to  $go_{root}.Info\_genes$ 
6:     end if
7:   end for
8:   if  $go_{root}$  has child GO node then
9:     for all  $go$ , such that  $go_{root} \rightarrow go$  do
10:      GO_gene_counter( $go, scores, \theta, \in, \leftarrow$ )
11:      Add unique  $go.Info\_genes$  to  $go_{root}.Info\_genes$ 
12:      Add unique  $go.Genes$  to  $go_{root}.Genes$ 
13:     end for
14:   end if
15: else
16:   return
17: end if

```

Algorithm 6 $bS(go, scores, G, \beta, \gamma)$: Best GO Adjusted Score

Require: go : array of GO nodes, $score(g)$: the single gene discriminative scores for each gene, G : list of genes, β, γ as defined before**Ensure:** bS : a list of best GO adjusted score for each gene in G

```

1: Initialize  $bS$  to be an associative array for each gene  $g \in G$  with minimum initial value.
2: for all gene  $g \in G$  do
3:   for all GO term  $go$ , such that  $g \in go$  do
4:     if  $|go.Info\_genes| > \beta$  and  $\frac{|go.Info\_genes|}{|go.Genes|} > \gamma$  then
5:        $S \leftarrow scores(g)$ 
6:     else
7:        $S \leftarrow 0$ 
8:     end if
9:     if  $bS(g) < S$  then
10:       $bS(g) \leftarrow S$ 
11:    end if
12:  end for
13: end for
14: return  $bS$ 

```

The algorithm to calculate GO adjusted scores is naturally divided into three steps. First step is to compute single gene-based discriminative scores whose complexity is not of interest here. The second step involves systematic traversal of a GO DAG. For each visitation of a GO node in the DAG traversal, we need first to analyze each gene that is annotated directly with such GO term, which takes $O(\mu)$ time; and then, we need to integrate gene lists from child GO terms into gene lists in current GO term, which takes $O(|G| \times \log|G| \times \xi)$ time in the worst case. Altogether step two takes $O(|GO| \times (\mu + |G| \times \log|G| \times \xi))$ in computation time. The third step examines each gene/GO term pair which takes $O(|GO| \times \mu)$ time. All these analysis are based on using the data structure in Fig. 4.2.1.

We would also like to consider complexity of building the initial data structure. GO flat file lists all GO terms one by one. Entry of a GO term also lists all its children GO terms. The time it takes to locate a GO term in the list of sorted GO terms is $O(\log(|GO|))$. For each GO term, ξ number of children GO terms needs to be located. The time it takes to build an GO DAG without annotations is $O(|GO| \times \xi \times \log(|GO|))$. The total number of GO annotation is $O(|GO| \times \mu)$. And the total time to insert GO annotations into our GO DAG is $O(|GO| \times \mu \times \log(|GO|))$. Thus the total time needed to initialize data structures used in our algorithm is $O(|GO| \times \log(|GO|) \times (\mu + \xi))$. This is an one time investment.

Space requirement for data structures mentioned in Fig. 4.2.1 is $O(|GO|)$ for the list of GO terms, $O(|GO| \times \mu)$ for the list of GO terms for each gene, $O(|G|)$ for the list of single gene discriminative scores and $O(|GO| \times |G| + |GO| \times \mu + |GO| \times \xi)$ for the GO DAG. The total space requirement for our algorithm is then $O(|GO| \times (|G| + \mu + \xi))$.

4.2.3 Experiment Setup

In this section, we describe the experimental structure we used. Our primary concern is the false discovery rate of any given single gene-based algorithm. We measure this false discovery rate by repeating experiments on randomized data set. Our experimental data set consists of the original data set from public available data sources and the random portion that are generated as noise. We measure for each single gene-based algorithm the

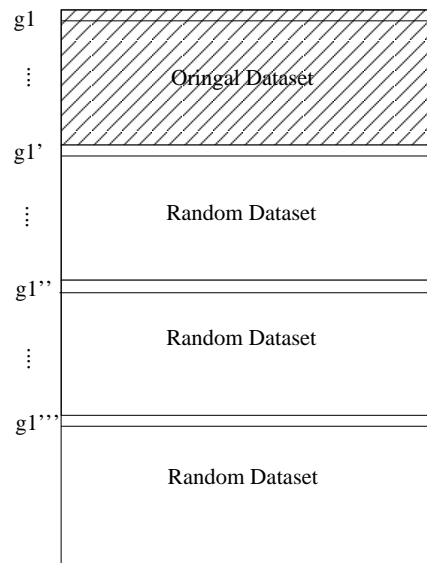


Figure 4.2.2: Setup of experiment data set for studying the alleviation of false positive rate on microarray expression data set.

percentage of genes that are selected coming from the randomized portion of data and use this number as an estimate of the false discovery rate.

As shown in Fig. 4.2.2, the original data set is appended with several repetition of blocks of random data. Each random block has the same number of genes and same number of experiments as the original data set. The expression levels in each random block is generated using normal distribution with same parameters as the original log transformed data set. Each gene in a random block corresponds to a real gene in the original data set and share the same GO annotations of it. For example in Fig. 4.2.2 the original data set is augmented by random data set three times its size. We call this experimental setup random sized 3. For each experimental setup, we generate 200 sets of random data and report the

average number of false positive.

4.3 Experiments

Data Preparation The data used in this work are all publicly available. We choose the widely used benchmark microarray data set: colon cancer [8]. For gene ontology, we downloaded a copy from GO web site at 10/15/2004. We collected GO annotation for genes used in colon cancer microarray experiment from SOURCE [32] online database on 11/1/2004. We do notice that SOURCE annotation may have been updated. However, the data set suffices to test our algorithm.

Not all genes in the original data set have GO annotation in SOURCE database. However, a decent majority of the original genes have been annotated. For colon cancer data set, out of original 2000 genes, we found 1495 of them have been annotated with at least one of the 9137 GO annotation. That is, on average, a little more than 6 GO annotation per gene. Since our algorithm relies on gene ontology to provide necessary biological insight, we choose only those genes that currently have GO annotation for further analysis. We expect more and more GO annotation will be available in the future.

Result Table 4.3.1 and 4.3.2 summarizes our experimental result on colon cancer data set. It shows the false positive rate measured as the number of random genes being chosen by four

single gene-based gene selection algorithms: S2N (signal to noise ratio) t-score and their GO adjusted counterpart. For each experiment we also show the number of overlapping genes, genes that are selected both both original single gene based discriminative scores and their GO adjusted counterparts. From this table, we observe that GO adjusted scores perform consistently better as less random genes (10%) are chosen by GO adjusted scores as informative genes. The trends shown in the tables are clear: the more random genes included the more false positive; the more number genes selected the more false positive. We generally have no knowledge and control of how many genes included in microarray data set are random with respect to sample class labels. However, we do know now that selecting excessive informative genes from microarray data set becomes troublesome. We also report the number of genes selected by both original discriminative score and the best GO adjusted score. The percentage of overlapping genes range from 60% to 80%, indicating majority of genes selected by original single gene based algorithms also cluster in GO annotation.

Now we describe how we choose the three parameters required by GO adjusted scores: θ, β, γ in these experiments. For θ , we set it to the cutoff score of selected genes. This essentially states that if user is to choose top 100 genes, we deem top 100 ranked genes as informative genes. Using the transitivity assumption of GO annotation, the virtual top GO node in our algorithm annotates every genes in question. Let the ratio between the number of informative genes and the number of genes in the virtual top GO node be γ_0 ,

Table 4.3.1: Performance of GO adjusted scores as measured in false positive rate using S2N score.

Random Size	# of Genes	S2N score	GO S2N score	Diff S2N score	Gene Overlap
1	50	1.40	1.15	0.25	42.45
2	50	2.40	2.15	0.25	39.05
3	50	4.15	3.75	0.40	39.85
4	50	4.05	3.55	0.50	28.64
1	100	5.30	4.09	1.21	68.63
2	100	8.78	7.57	1.21	68.06
3	100	17.24	14.84	2.40	56.38
4	100	15.02	12.55	2.47	59.63
1	200	19.10	17.45	1.65	167.85
2	200	33.15	30.30	2.85	153.90
3	200	43.10	39.45	3.65	144.10
4	200	49.60	46.90	2.70	132.65

experiments show good results when γ is set to 1.4-1.6 times γ_0 . When γ is set to γ_0 , GO adjusted scores revert to original single gene-based discriminative score since the virtual top GO node becomes informative. With the increase in γ , we are taking information in GO annotation more assertively. In our experiments β is set to 1. We tested β values from 0 to 2, results are similar.

4.4 Conclusion and Future Work

Gene selection plays an important role in the analysis of microarray data set. Genes that express differently in different sample conditions are selected for further biological investigation. However, due to the limited sample size and complex underlying biology, gene

Table 4.3.2: Performance of GO adjusted scores as measured in false positive rate using t-score.

Random Size	# of Genes	t score	GO t score	Diff t score	Gene Overlap
1	50	1.40	1.20	0.20	41.85
2	50	2.90	2.25	0.65	38.60
3	50	3.95	3.3	0.65	38.10
4	50	4.90	4.25	0.65	38.55
1	100	5.15	4.26	0.89	66.28
2	100	9.06	7.38	1.68	74.06
3	100	18.11	16.49	1.62	60.14
4	100	15.58	13.20	2.38	67.76
1	200	19.40	17.60	1.80	166.35
2	200	31.95	30.15	1.80	155.30
3	200	44.70	42.30	2.40	136.75
4	200	53.35	49.65	3.70	129.45

selection algorithms are haunted by excessive false positive rate. In this work, we proposed to integrate biological domain knowledge imbedded in gene ontology and its annotation into the process of gene selection. This is the first attempt of such integration, to our best knowledge. Our experimental result shows this is a promising direction. Using gene ontology and its annotation, the probability of selecting random genes in our experiments is reduced more than 10% on average. Our algorithm is a wrapper algorithm upon any if not all single gene based discriminative scores. Our algorithm behaves differently with different choice of one parameter γ , taking GO annotation into consideration within the feature selection process at various degrees. This provides an interesting way to integrate “old” knowledge in GO ontology with “new” information from microarray experiments.

We ignored genes without GO annotation in this work for simplicity. Although majority of

genes (75%) in our study are annotated by at least one GO term, genes that are not currently annotated may be of interest nonetheless. For these genes, traditional discriminative scores can be used instead. Best GO adjusted score defined by Def 4.2.5 is comparable to original discriminative scores since they are essentially the same if the annotating GO term is informative. A straightforward way to handle gene selection for genes that are not currently annotated would be to compute original discriminative scores. Such discriminative scores can then be combined with best GO adjusted scores for the purpose of gene selection.

The discriminative power of a GO term is defined to describe the correlation between GO terms and sample class labels. This is different from the previous research in the correlation between gene expression similarity and GO annotation similarity [11, 124], which has some interesting implication by itself. It provides a bridge between gene ontology and disease ontology. The ability of coupling GO terms and disease symptoms may prove useful to provide biologist new insight into disease pathology.

Some recent research [104] has focused on integration of different genomic data sets for inferring pathways and regulative networks. However we have not found much work on integration of genomic data sets for gene selection. Our work naturally extends to the idea of integration of other genomic data sets for the purpose of target selection. This is the future work we will actively pursue.

This chapter represents our first attempt to integrate gene ontology and its annotations into

feature selection process for microarray expression data set. In the next chapter, we extend virtual gene approach using gene annotations. Virtual genes now are not limited to pairs as in Chapter 3. Encouraging results are achieved using certain branch of GO structures.

Chapter 5

Integration of Gene Ontology with Virtual Gene

In the previous chapters, we examined the use of gene ontology to reduce false positive rate problem facing gene selection algorithms with small sized samples. We also examined selecting informative genes based on gene correlations. More specifically, in Chapter 3 In this chapter, we present a novel feature extraction algorithm based on the concept of virtual genes by integrating gene expression data sets with domain knowledge imbedded in gene ontology annotations. GO distance is defined between a pair of genes based on their annotating GO terms and the relationships between those GO terms. Groups of genes that are similar to each other based on their GO annotations are identified. Correlations in gene

expression levels within those groups of genes (virtual genes) are then analyzed and used to build better sample classifiers. Unlike other feature extraction techniques such as PCA and SVD, each virtual gene corresponds to a linearly combined collection of real genes and to some extent preserves their original meaning.

This chapter is organized as follows. In Section 1, we introduce a new way to measure distance between genes based on their GO annotations. The new feature extraction algorithm that integrates GO annotation distance is proposed in Section 2. Experimental result is provided in Section 3. In Section 4, we conclude this chapter.

5.1 The Hierarchical Structure in Feature Space

In this section, we will explain the hierarchical structure that exists for genes in term of Gene Ontology (GO) and GO annotations. We will define how to measure the specificity of GO terms and how to measure distance between genes based on their GO annotations.

In certain application domains, there exist hierarchical structures in the feature space. For example, in content based image retrieval, features can be classified either as colors or as textures. Genes are considered features of a microarray expression data set for the purpose of sample classification. Each gene on the other hand is annotated by some GO terms. The databases of GO annotations store an important part of our current biological knowledge

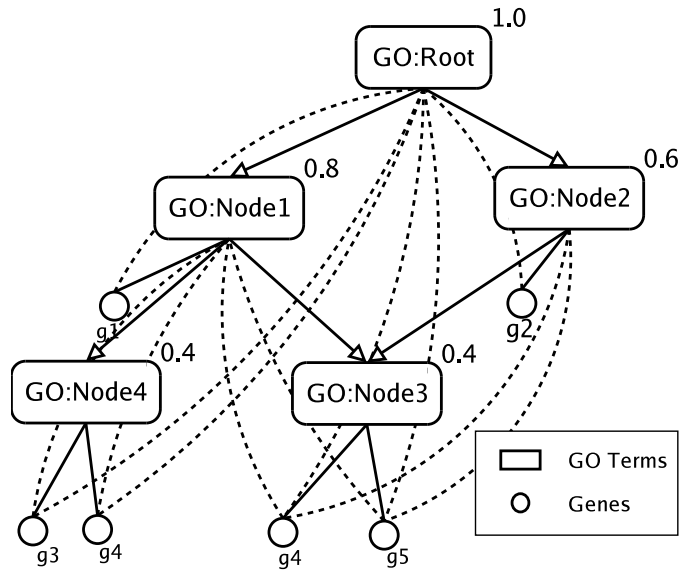


Figure 5.1.1: Structure of gene ontology and its annotation. Solid lines between genes and GO terms indicate direct annotation and dotted lines indicate inferred annotation by the property of transitivity of GO annotations.

of the genes and their products under investigation.

The Gene Ontology (GO) project is a collaborative effort to address the need for consistent descriptions of gene products in different databases [28]. Gene Ontology (GO) is first a collection of shared biological terms so that biologists of different branches could share the vocabulary. It also specifies certain relationships between terms such as “is-a” and “part-of” relationship. The GO terms and their relationships form a DAG (directed acyclic graph) [29] as indicated in Figure 5.1.1. The GO collaborators are developing three structured, controlled vocabularies (ontologies) that describe gene products in terms of their associated biological processes, cellular components and molecular functions in a species-independent manner [28].

Definition 5.1.1 *The coverage of a GO term is defined as the percentage of genes, out of all genes involved in one study, a GO term annotates.*

Definition 5.1.2 *The GO distance of two genes is defined to be the coverage of their most specific common annotating GO term.*

Definition 5.1.3 *The gene distance graph is an undirected weighted graph $A(V_a, E_a)$ where vertices V_a is the set of genes in a study and edges E_a is the GO distance between corresponding vertices (genes).*

Definition 5.1.4 *The gene connectivity graph $B(V_b, E_b)$ is an undirected unweighted graph inferred from $A(V_a, E_a)$ and a parameter min_e , where $V_b = V_a$ and $(v_1, v_2) \in E_b$ if and only if $E_a(v_1, v_2) < min_e$. $E_a(v_1, v_2)$ is the edge weight of (v_1, v_2) in graph A .*

Figure 5.1.1 illustrates the structure of five GO terms and five genes which are annotated by some of those five GO terms. Although simple, it captures the major structural elements in GO and GO annotations: 1. GO terms form a DAG; 2. each gene is annotated by one or more GO term(s). E.g. in Figure 5.1.1 GO:Node3 has two parent GO terms, GO:Node1 and GO:Node2. Gene g_4 is annotated by both GO:Node3 and GO:Node4.

The relationship between GO terms is part-to-whole or specific-to-general. It is reasonable to assume when a gene is annotated by a GO term, it is also annotated by all the more general GO terms on the path from that GO term to the root in the GO DAG structure. We

call this property of transitivity of GO annotations. In Figure 5.1.1 solid lines are used to connect genes to their direct annotating GO terms. On the other hand, dotted lines are used to show the annotations inferred using the transitivity property.

The specificity of a GO term measures how broad the concept associated with the corresponding GO term is. How do we measure the specificity of a GO term? In this chapter, the coverage of a GO term is defined to be the percentage of genes it annotates out of all genes in question, including both direct annotations and annotations inferred from transitivity property. The smaller the coverage of a GO term is, the more specific the concept associated with the GO term is. The root (virtual) GO term has the largest coverage value, which is 1 in case all genes in a study have some GO annotations (the root GO term annotates every gene). The coverages of GO terms from root to leaf form a non-increasing sequence, ranging from 1 to 0. The numbers in the top right corners of GO nodes in Figure 5.1.1 indicate the coverage of the corresponding GO term in our example.

Genes that are annotated by the same GO terms are similar to each other in some biological aspects to a certain degree. Given a set of genes that are annotated by the same GO term, the more specific this GO term is, the more specific biological processes, cellular location or molecular functions this set of genes share. Using the definition of GO term coverage and GO annotations of genes, we can gauge how similar two genes are based on their GO annotations. The GO distance between two genes are formally defined in Definition 5.1.2.

Two genes are close to each other in GO distance if they are both annotated by some specific GO terms. We use the coverage of the most specific common annotating GO term as a measurement of distance between two genes based on information imbedded in GO annotations. Continue with our running example in Figure 5.1.1, the GO distance between genes g_3 and g_4 is 0.4 and the GO distance between g_2 and g_4 is 0.6. We further define the GO diameter of a set of genes to be the maximum pairwise GO distance among this set of genes.

The genes under investigation in a study and their GO distances form a graph, which is referred to as the gene distance graph in this chapter. The vertices of this graph are genes and edges between vertices are the GO distances between corresponding pair of genes. For simplicity, gene distance graph is thresholded by a parameter min_e . Edges with weight larger than min_e are removed from gene distance graph, resulting in gene connectivity graph. A clique in a graph is a full connected subgraph. Within gene connectivity graph, each clique identifies a tightly related set of genes, in respect to the GO distances. Such cliques found in the gene connectivity graph are further referred to as gene cliques in this chapter. Parameter min_e is mainly used to control the size of each gene clique. We choose to control the maximum size of gene clique in this chapter to be around 20.

Gene clique has a special role in this chapter. It identifies a set of biologically related genes, genes that participate in related biological processes or exhibit similar molecular function

or exist in similar cellular components. The evidences supporting such clustering all come from gene annotations imbedded in online databases.

5.2 Feature Extraction by Integrating Ontology Distance with Gene Expression Data Set

As mentioned before, gene selection as a process to identify the best subset of genes for sample classification tasks is exponentially hard. Traditional feature selection algorithms ignore the correlation between genes. However, such correlation is an integral part of the underlying biological system. Genes collaborate with each other in various biochemical pathways. In fact, the common assumption underpinning clustering analysis of gene expression data is that genes with similar functions tend to expression similarly [70]. We have shown in [134] that pairwise correlation between genes could be used to construct meaningful classifiers.

Single gene based discriminative scores and exhaustive search of the power set of genes can be viewed as two extremes in the field of feature selection. On one hand, single gene based discriminative scores ignore all interactions between genes; on the other hand, exhaustive search of the power set of genes examines all possible interactions among all genes, no matter whether there is any biological evidence supporting the existence of such interaction.

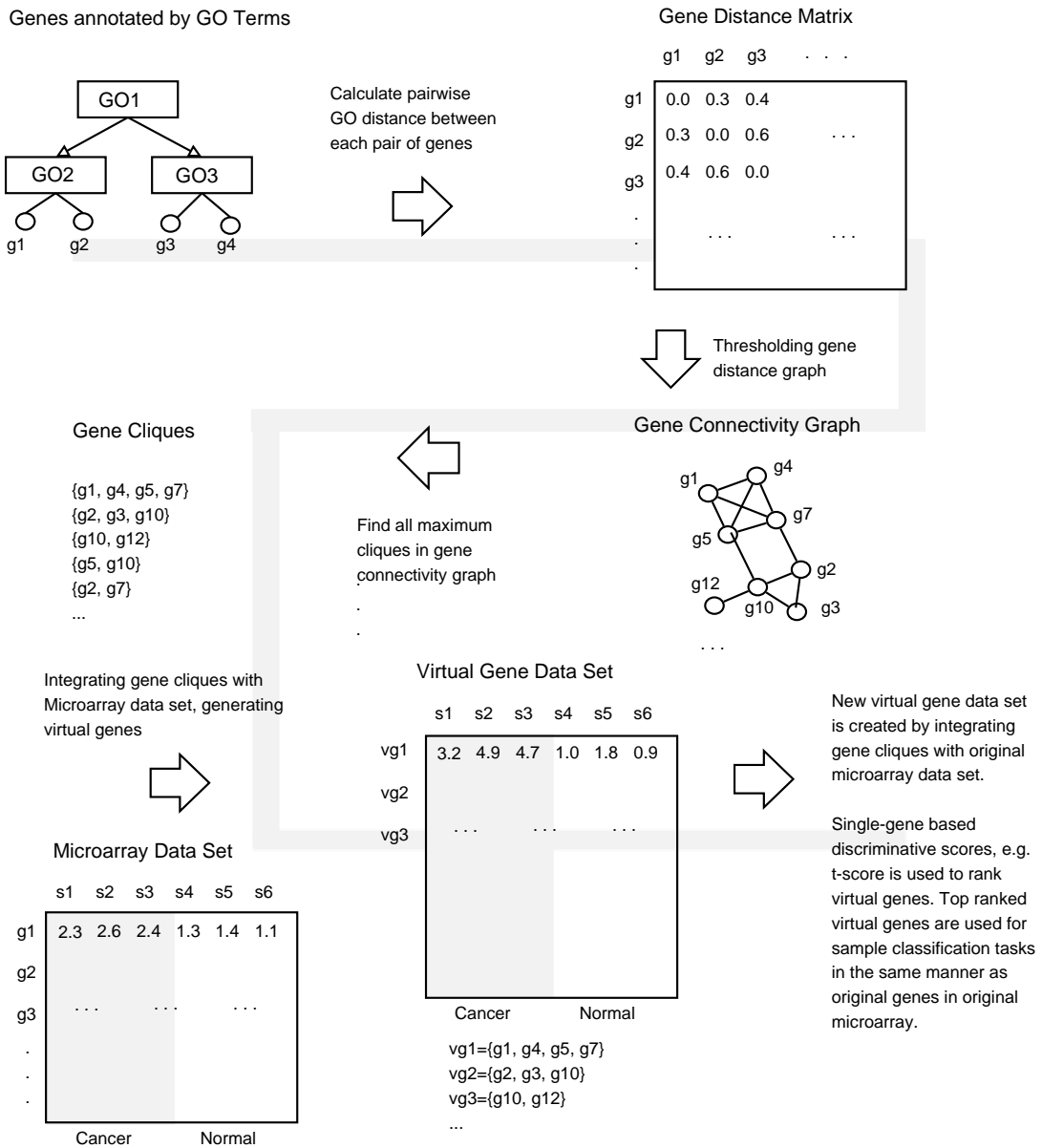


Figure 5.2.1: Flowchart of our virtual gene feature extraction algorithm for sample classification on microarray data set by integrating domain knowledge in the form of GO annotations.

Neither one is a sound solution. Some other feature selection algorithms sit in the middle of these two extremes. The float selection (SFFS) algorithms could be viewed as compromises that examine a selected subset of feature (genes) by adding or removing one gene at a time. Although there is not much biological justifications of the selection of such gene subsets.

Is there more efficient way to guide the search between these two extremes? For biological systems under investigation, we observe the following characteristics:

- The correlation between features is an integral part of information of the underlying system. Ignoring any interaction between genes has to be an oversimplification.
- The number of genes in typical biochemical pathways are limited to a rather small number. Investigating correlations among a big set of genes not only increases computation exponentially, but also invites unwanted noise.

In this chapter, we propose to use GO and its annotations to guide this search process. Using the GO distance defined in the previous section, genes that are more similar to each other based on GO annotations are further examined whether certain linear combination of them serves a good predictor of sample class labels. If there exists such a linear combination, we further create a new “feature” called virtual gene to capture it. The sample classification algorithm later uses those virtual genes instead of original genes as features describing molecular states of samples under investigation.

In this section, we will first introduce the concept of virtual gene and virtual gene expression we used in an earlier chapter [134]. We copied some definitions here for easy reference. A virtual gene is a group of genes that work collectively in the eye of a classifier. Thus virtual genes can be viewed as new features extracted from the original microarray data set. It consists of the constituent gene set and parameters to combine them together. Given a virtual gene and a set of gene expressions of the constituent genes, virtual gene expression can be derived according to Definition 5.2.2. The parameters of a virtual gene represent the best linear direction to which the constituent genes can be best combined to form a linear classifier distinguishing different sample class labels. We used FLD (Fisher Linear Discriminant) [127] to find such best linear direction. For details of the virtual gene algorithm please refer to our earlier chapter.

Definition 5.2.1 *Virtual Gene* is a triplet $VG = (G_v, W, b)$ where $G_v \subseteq \mathcal{G}$ is a set of constituent genes, \mathcal{G} is the set of genes under study, W is a matrix of size $|G_v| \times 1$, b is a numeric value. The expression levels of a **virtual gene** is determined using Definition 6.

Definition 5.2.2 (Virtual Gene Expression) Given a **virtual gene** $VG = (G_v, W, b)$ and gene expression matrix E , where $|G_v| = n_v$, E is an $n_v \times m_v$ expression matrix, the **virtual gene expression** VE of a virtual gene VG is a linear combination of expression matrix E . $VE(VG, E) = W' \times E + b$, where W' is the transpose of W .

Figure 5.2.1 shows the flow of our virtual gene feature extraction algorithm. Overall, our

algorithm works in three stages. The first stage generates possible gene subsets of interest with GO diameter less than or equal to some predefined parameter min_e , which are the gene cliques mentioned earlier. Biological knowledge imbedded in the GO annotations is abstracted into the GO distance measurement between genes. Treating each gene clique as constituent genes, a virtual gene is created for each gene clique in the second stage by integrating it with the microarray gene expression data set. Virtual gene expressions are also computed in this stage. Knowledge integration is archived in the second stage. The final stage uses a single gene based discriminative score and selects the best virtual genes from the pool of virtual genes. We used the common t-test score in stage 3 for this chapter, although other implementations are possible too.

The first stage of our algorithm generates gene cliques used in the later stages of our algorithm. It starts from the GO DAG with genes attached to their annotating GO terms. Pairwise GO distance is computed for every pair of genes, resulting in the gene distance graph (matrix). Then the gene distance graph is truncated using parameter min_e into an unweighted gene connectivity graph. Two genes are connected in the gene connectivity graph if and only if their GO distance is smaller than min_e . Now we are finally ready to compute the maximum gene cliques from the gene connectivity graph.

There are two intensive computational problems involved in this stage. The first is to calculate the gene distance graph, which involves computing all pairwise GO distances between

genes. The second intensive computation is to find cliques in the gene connectivity graph. Finding maximal cliques has been proven to be NP hard problem. In our case, we have limited the connectivity of our graph using a truncating factor min_e . The resulting gene connectivity graph normally contains cliques with less than 20 genes. This makes a contemporary PC capable of handling the clique finding problem. The truncation of the gene distance graph is reasonable since the number of genes that tends to interact with each other is small, based on other biological evidence. We used an algorithm similar to [21] to find all cliques in our gene connectivity graph.

In the second stage, a new feature called virtual gene is created for each gene clique in the gene connectivity graph by integrating it with the microarray data set. Given a set of genes and their expression levels on different sample classes, genes are used as features describing molecular states of samples. FLD (Fisher Linear Discriminant) [127] is used to identify the best linear direction that separates sample class labels. Genes are used as features. Each sample becomes a data point in a multi-dimensional space. Such data points are labeled by sample class labels. FLD computes the best linear direction in this multi-dimensional space such that projection of data points onto this direction produces best linear classifier of two class labels. If there is only a single gene in a gene clique, the real gene is used in the place of a virtual gene. The parameters such as the best linear direction and the intersect are stored for each virtual gene so that virtual gene expression could be computed for testing data sets.

In the final stage, some single gene-based discriminative score, such as t-test score, is calculated for every virtual gene. Such scores are sorted and the top ranked genes (virtual or real) are used for the purpose of sample classification tasks.

5.3 Experiments and Discussion

Extensive experiments are performed on two publicly available data sets: colon cancer data set [8] and leukemia data set [53]. Gene annotations are extracted from the SOURCE [32] database using the gene bank accession id provided in the original papers. Colon cancer data set contains measurements of expression levels of 2000 genes over 62 samples, 40 samples were from colon cancer patients and the other 22 samples were from normal tissues. Leukemia data set consists of 7129 genes and 72 samples, of which 47 samples were acute lymphoblastic leukemia (ALL) and rest 25 samples were acute myeloid leukemia (AML).

Gene ontology is categorized into three rather distinct structures: biological process (BP), molecular function (MF) and cellular component (CC). We performed experiments separately using each of those structures. Not all genes used in the original microarray studies are annotated by some GO terms. In order to compare performance of our GO based algorithm fairly with other approaches, we only use those annotated genes for other feature

selection/extraction methods. This requirement results in 1253 genes for cellular component ontology, 1364 genes for biological process ontology and 1388 genes for molecular function ontology on colon cancer data set; 3936 genes for cellular component ontology, 4317 genes for biological process ontology and 4282 genes for molecular function ontology on leukemia data set. Thus six different data sets are experimented in total.

The performance of feature selection (extraction) algorithms are measured using the sample classification performance of three vastly different classifiers: DLD (Diagonal Linear Discriminant) [83], KNN (K-Nearest Neighbor, with $k=3$) [87] and SVM (Support Vector Machine with radial kernel) [23]. DLD is a linear classifier. KNN is an instance-based nonlinear classifier. And SVM is arguably the most popular general purpose classifier. The vast difference of the three chosen classifiers ensure any performance difference we observe in sample classification accuracy can be properly attributed to the different feature selection/extraction methods used. We programmed all the experiments in R [52].

We use a two-fold cross validation scheme to estimate the classification accuracy. In our experience, two-fold cross validation produces results with less variance as compared to the more usually used leave-one-out cross validation. Less variability means more statistically significance for the comparison of results. Two-fold cross validation works as follows: the samples available are divided into two roughly equal sized groups. The numbers of samples of different sample classes are also kept roughly identical for each groups. One

of the two groups is used for training and the other is used for testing. Genes are first selected/extracted from the training data set. Then classification performance of the three selected classifiers are measured using those genes or gene extraction models on the test data set. We randomly repeat this two-fold cross validation setup for 100 times and the average performance is reported here in Table 5.3.1 and Table 5.3.3.

Genes with GO annotations are further filtered to remove genes without significant changes across samples. Gene expression data in colon cancer data set is well formatted and thus not filtered. Gene expression data in leukemia data set contains excessive negative values. Using criteria similar to [14], we first threshold the data to the range from 0 to 16000. Further, genes with expression level change less than 5 fold or absolute change less than 500 are removed. Resultant gene expression levels are log transformed and normalized before used in our experiments.

We experimented with three gene selection and extraction algorithms: t-test score [15], signal-to-noise ratio (S2N) [8] and our GOF (Gene Ontology based Feature extraction). T-test score and S2N do not use gene annotations. In order to compare fairly with gene ontology based algorithms, we only use those genes that are annotated for t-test score and S2N. This results in six different data sets for the two data sets we used in our experiment: colon cancer, leukemia combined with the three gene ontology structures: molecular function, biological process, and cellular component.

Table 5.3.1: Performance (classification accuracy) of GO based virtual gene feature extraction algorithm (GOF), compared with single gene based algorithms: t-test score (tscore) [15], signal-to-noise ratio (S2N) [53], on colon cancer data set. Numbers in parenthesis are the number of genes. *vg* stands for virtual gene and *rg* stands for real gene. We also report classification performance when no feature selection algorithm is used (ALL column).

GO (# genes)	Class. alg.	ALL %	tscore % (# genes)	S2N % (# genes)	GOF % (# of <i>vg/rg</i>)
CC(1253)	DLD	63.8	75.3(12)	75.4(12)	86.9 (31/298)
	KNN	77.0	82.3(82)	81.9(92)	85.5 (71/389)
	SVM	70.5	79.8(52)	80.0(62)	85.0 (31/298)
BP(1364)	DLD	63.4	77.5(12)	76.7(12)	86.6 (131/634)
	KNN	77.1	83.0(92)	82.5(82)	83.2(191/695)
	SVM	70.5	80.8(62)	80.4(32)	85.4 (171/677)
MF(1388)	DLD	64.7	77.4(12)	76.9(12)	83.5 (81/682)
	KNN	78.1	83.1(122)	82.6(82)	81.2(191/886)
	SVM	70.9	80.0(82)	80.6(82)	81.2(61/609)

Table 5.3.2: Standard deviation of classification accuracy of GO based virtual gene feature extraction algorithm (GOF), compared with single gene based algorithms: t-test score (tscore) [15], signal-to-noise ratio (S2N) [53], on colon cancer data set.

GO (# genes)	Class. alg.	ALL %	tscore %	S2N %	GOF %
CC(1253)	DLD	13.2	7.6	7.6	4.2
	KNN	6.3	7.5	7.8	5.5
	SVM	4.4	8.0	8.1	5.6
BP(1364)	DLD	13.1	8.3	8.6	4.3
	KNN	6.2	7.2	7.2	5.9
	SVM	4.5	7.7	7.6	5.1
MF(1388)	DLD	12.8	8.4	8.4	5.1
	KNN	6.2	6.7	6.6	6.3
	SVM	4.6	6.7	7.0	5.1

Table 5.3.1 and Table 5.3.3 summarize the classification accuracy for three feature selection (extraction) algorithms: t-test score, S2N and our GOF (Gene Ontology based Feature extraction). Also showed in parenthesis are the genes (virtual or real) used in each scenario when applicable. For single gene based discriminative scores, the number represents the number of genes selected that results in best classification performance. For our GOF algorithm, both the number of virtual genes and the number of real genes that produce those virtual genes are reported. We used bold font to distinguish those cases where GOF outperforms other algorithms significantly. Since we do not have prior knowledge of how many genes should be chosen, the best classification accuracy and the number of selected genes in those best cases are reported here.

On colon cancer data set (Table 5.3.1), when cellular component ontology is used, GOF performs consistently and significantly better than the other two gene selection algorithms. The performance increases ranging from 5% to more than 10%. Biological process ontology is also useful on colon cancer data set, with the exception of KNN classifier. On the contrary, molecular function ontology is only useful for DLD classifier. It is also worth noting that the variance of the classification accuracy for GOF is much smaller than those of the two single gene-based algorithms in our experiment, as reported in Table 5.3.2. From those experiment, we conclude the gene groups associated with the cellular component ontology contains valuable information of the distinction between colon cancer and normal tissues. Further biological investigation is needed to further our understanding in this area.

On leukemia data set ¹, we get more mixed result. GOF outperforms single gene based algorithms when SVM classifier and cellular component/molecular function ontology are used. One explanation is that the classification performance of all three classification algorithms are already pretty good on leukemia data set, ranging in the mid-90s. The room for improvement is quite limited.

There is one parameter min_e in our algorithm, which is used to control size of gene cliques. The reason to control the size of gene clique is two-fold. First of all, computing maximum cliques of a graph with thousands of nodes is only feasible if the clique size is kept small. Secondly, genes tend to form small groups rather than large groups in biological processes. In our experiment, we choose min_e to be 0.5% to 1%, keeping the size of maximum gene cliques to be around 20.

5.4 Conclusion

In this chapter, we proposed a novel feature extraction algorithm based on the concept of virtual gene by integrating gene expression data set with domain knowledge imbedded in gene ontology annotations. In order to do this, we first proposed a new way to measure distance between genes based on their gene ontology annotations, namely GO distance. Genes

¹DLD classifier failed because the data is exactly singular when all genes are used.

Table 5.3.3: Performance (classification accuracy) of GO based virtual gene feature extraction algorithm (GOF), compared with single gene based algorithms: t-test score (tscore) [15], signal-to-noise ratio (S2N) [53], on leukemia data set. Numbers in parenthesis are the number of genes. *vg* stands for virtual gene and *rg* for real gene. We also report classification performance when no feature selection algorithm is used (ALL column).

GO (# genes)	Class. alg.	ALL %	tscore % (# genes)	S2N % (# genes)	GOF % (# of <i>vg/rg</i>)
CC (3936)	DLD	NA ¹	96.5(202)	96.2(152)	96.4(111/528)
	KNN	92.8	95.9(392)	96.4(362)	95.5(231/668)
	SVM	89.3	93.5(22)	94.7(392)	96.2(51/428)
BP (4317)	DLD	NA ¹	96.4(142)	96.4(92)	95.8(241/1049)
	KNN	92.0	95.9(392)	96.2(392)	94.2(71/638)
	SVM	88.5	93.5(22)	95.5(252)	93.0(121/827)
MF (4282)	DLD	NA ¹	96.3(92)	96.1(112)	96.8(261/1235)
	KNN	91.6	96.1(382)	96.1(382)	95.4(291/1280)
	SVM	88.1	92.7(32)	95.1(252)	95.0(251/1220)

Table 5.3.4: Standard deviation of classification accuracy of GO based virtual gene feature extraction algorithm (GOF), compared with single gene based algorithms: t-test score (tscore) [15], signal-to-noise ratio (S2N) [53], on leukemia data set.

GO (# genes)	Class. alg.	ALL %	tscore %	S2N %	GOF %
CC(1253)	DLD	NA ¹	2.4	3.9	2.5
	KNN	3.6	2.7	2.5	2.6
	SVM	4.4	3.7	4.6	3.1
BP(1364)	DLD	NA ¹	2.4	2.3	3.0
	KNN	3.8	2.6	2.8	3.7
	SVM	4.3	3.6	4.2	4.2
MF(1388)	DLD	NA ¹	2.2	3.9	2.7
	KNN	3.7	2.9	2.8	3.1
	SVM	4.4	3.6	4.5	4.0

are then clustered based on such GO distance. Each of the resulting gene cluster (clique) is analyzed whether it is a good group of genes whose expression levels separates sample class labels well. Each of such clusters is transformed into a virtual gene. Top ranked virtual genes are then selected for sample classification. We experimented our algorithm on two publicly available data sets. Consistent and significant improvement is achieved on colon cancer data set using cellular component ontology. This demonstrates the benefit of integrating domain knowledge information with the gene expression data set.

This is our second attempt trying to integrate gene ontology annotations directly in the gene selection (extraction) process. Currently a virtual gene is created for each gene clique. It also makes sense to run wrapper algorithms within each gene clique to identify gene subgroups that explain sample variation well. Since each gene clique is of limited size by controlling parameter min_e , it will not be prohibitively expensive to do an exhaustive search within gene cliques. The obstacle facing us right now is how to decide which and how many gene subgroups within each gene clique to choose for sample classification. We are currently examining this topic.

During the actual classification stage, we only used virtual gene generated from gene cliques. As there's no fundamental difference between virtual gene expression levels and real gene expression levels. We could have attempted to just combine the virtual gene data set with original data set and run single gene based feature selection algorithm. The result

is a mixture of virtual genes and real genes. The problem with this approach is that virtual genes normally have higher discriminative power. Some balance needs to be sought between virtual genes and real genes.

Chapter 6

Conclusion of the Thesis

Bioinformatics as a new and active research domain, poses a lot of new and interesting problems for both computer scientists and biologists. It encompasses biology, statistics and computer science and examines the fundamental question of life on earth.

Owing to the rapid advances in biological techniques, large scale profiling of RNA/protein levels produces enormous amount of data, far beyond the processing capability of any single person. Computational algorithms are needed to find patterns in those data sets and draw valid biological conclusions in order to advance our understanding of living organisms.

Microarray gene expression data set under investigation in this thesis is one of the first

RNA large scale profiling data set available. It provides us for the first time in history a peek into molecular states of tissue samples in question. Although mRNA level is not a direct indicator of protein level due to various post-transcriptional modification, it provides a rough indicator of it.

During our study on microarray gene expression data set, we observed the following characteristics for this kind of data set.

- The dimensionality of the data set is ill formed for machine learning and pattern recognition algorithms. More precisely, the abundance of features and scarcity of samples makes classification algorithm less effective. It also amplifies noise, making computational result less statistically significant. As a comparison, a benchmark microarray expression data set [8] consists of 2000 genes (features) and 62 samples; while of the 27 data sets studied in a classic paper in machine learning literature [47], the number of training examples ranges from 47 to 20000 and the number of features used ranges from 4 to 69. Feature selection/extraction algorithms are direly needed in the bioinformatics field in order for the learning to be effective.
- Unlike widely used assumption of independence between features, features (genes) are correlated with each other through various biological pathways. Could the correlations among features be used for the purpose of sample classification?
- The expected number of genes interacting with each other is expected to be small.

Interacting genes and proteins normally form small groups. This derives from the fact that pathways and interacting protein groups tend to be small.

- The microarray expression data set itself is not isolated. There are a plenty of domain knowledge on each features (genes) and samples (experiments). There are a plenty of other kinds of data sets that uncover other aspect of underlying biological processes. Different data sets can be used to reinforce each other. More specifically, there are a lot of domain knowledge about the genes in microarray expression data set.

During our study, we identified the deficiency of current gene selection/extraction algorithms in the literature as follows:

- Correlations among genes are often ignored. Most previous gene selection approaches are single gene-based. These methods simply assume genes are independent while ignoring the correlation between genes.
- Microarray expression data set is used alone for sample classification. Although there is a plenty of relevant information, not much research has attempted to integrate it with microarray expression data set for better sample classification.

In this thesis, we examined how to build effective classifiers for sample classification problems using microarray expression data set. We focused on feature selection/extraction algorithms that are tailored to those characteristics of gene expression data set mentioned

above. Four novel feature selection algorithms are proposed in this thesis to address different aspects of gene selection/extraction tasks. We summarize them here as the follows:

- We first examine correlation between genes using permutation based method (BFSS: Boost Feature Subset Selection [136]). Subsequential genes are selected based on those genes that have previously been selected, emphasizing on previously difficult samples.
- Continue to examine correlations between features, we proposed the concept of virtual gene that explicitly measures correlations between a group of genes and sample classes. The combined expression levels, or virtual gene expression levels, are used for sample classification. Even pairwise virtual gene shows improvement over single gene-based gene selection algorithms [134].
- In [132], we examined the integration of gene ontology annotations with microarray expression data set, in the context of reducing false positive rate. We showed by integrating GO annotation, genes selected are less likely coming from randomness in their expression.
- In the final chapter of this thesis, virtual gene algorithms is expanded with the integration of gene ontology annotations. We used GO annotation to generate a categorization of genes. Genes are grouped based on such categorization. This enables us to go beyond pairwise virtual gene. Bigger gene groups (usually up to 20 genes) are

examined for meaningful correlations. Significant improvement in sample classification accuracy is observed in some cases.

BFSS (Chapter 2) is a novel general feature subset selection framework to improve the performance of single-feature based discriminative scores. Genes (features) are selected from bootstraps of training set instead of training set itself. The sampling probability is dynamically adapted based on the performance of previously selected genes on different bootstrap samples. In this way, the gene selection process is tilted toward those samples that previously selected genes work less than satisfactory. A nice feature of our approach is that most if not all single-gene based discriminative scores can be plugged into our system and the resulted BFSS feature selectors are expected to perform better than the original scores according to our experiments.

As shown in the case of BFSS, considering correlation between genes is beneficial for the task of feature selection. In Chapter 3, we proposed virtual gene that examines pairwise correlations between every gene pairs. Instead of trying to minimize correlations within the selected gene set as some research in the literature did, we examined whether such correlations are good predictors of sample class labels. Virtual gene is a linear combination of a set of real genes. Our experiments confirm our assumption that the correlations between genes are indeed good predictors of sample class labels, better in many cases than single gene based discriminative scores. However, in Chapter 3, we have limited the size of virtual

gene to 2 (pairwise) because of computation cost of bigger virtual genes. Such constraint is circumvented in Chapter 5 by using heuristics from domain knowledge. In Chapter 5, biologically related genes are examined for correlation.

BFSS and pairwise virtual gene algorithms do not integrate domain knowledge. However, domain knowledge is always an important factor in any branch of science. It is extremely so for biology. First step has been taken to make such knowledge accessible to computer algorithms. Gene ontology [28] is an effort to create a common vocabulary that biologists working on different organisms can share. GO annotations use GO terms to label genes and their products. Online databases such as NCBI [91] become new hubs for scientists working in bioinformatics to search for relevant information.

We further investigated how to integrate domain knowledge into the feature selection/extraction process for the special problems in bioinformatics domain. In Chapter 4, gene annotations are used to alleviate the false positive rate problem for gene selection on microarray expression data set by modifying single gene-based scores using “GO adjusted scores” and “best GO adjust scores”. Gene selection on microarray expression data generally suffers the false positive rate problem because of the skewed dimensionality of the data set itself. Even randomly generated features of this size will contain some “significant” genes. In our algorithm, genes with similar annotations reinforce each other’s scores. The feature

selection process is thus tilted toward well known gene clusters that have been well annotated. We actually betted on those random genes are not likely to be clustered with same GO annotations. This is our first attempt to integrate gene annotations. The “GO adjusted scores” are defined rather coarsely using a “take it all or leave it” formula.

Gene annotation is also used in Chapter 5 to enable virtual gene algorithms to go beyond gene pairs. The basic idea of virtual gene algorithm is to use combined expression levels of virtual genes for classification. The problem is which group of genes to choose as the constituent genes for a virtual gene. As our second attempt of knowledge integration with gene expression data set, a novel and more robust gene distance called GO distance is defined using GO structures and gene annotations. Two genes are similar if they are annotated by some common specific GO terms. By “specific”, we mean the less number of genes that are annotated by a GO term, the more specific this GO term is. Because of the transitivity of GO annotations, this generally means the deeper a node in GO hierarchy, the more specific it is. By our definition, two genes are similar if they appear in rather confined (specific) cellular locations, for example, when cellular component ontology is used. Genes are clustered based on their GO similarities. The resulting gene groups are called gene cliques. Each gene clique forms a virtual gene. The correlations of each virtual gene is then examined and top ranked virtual genes based on some single gene-based discriminative scores are then selected for sample classification. One drawback of our GO distance is that it is not a metric. We have to use clique finding algorithms to find gene clusters based on

this similarity. If we can make GO similarity a metric, the whole virtual gene algorithm could be significantly sped up. Our experiments confirmed that using proper branch of GO annotations, significant gain in sample classification accuracy can be obtained.

Feature selection and extraction algorithm is the first and an important step in designing a successful machine learning system. Although we are dealing with the rather specific microarray gene expression data set in this thesis, the basic discoveries in this thesis are applicable to other domains where machine learning touches as well. Our discoveries can be condensed into two points. 1. Feature correlations could be important information we have on sample class labels. 2. Different types of data sets in a domain contain different information of the underlying system, intelligent integration of different data sets proves to be very useful in this thesis.

Those principles are understood and used in various research domains. For example, in content based image retrieval (CBIR) systems, images are not represented using the native RGB pixel values. Rather, various manmade features are created, such as color histogram, edge texture, etc.. Those manmade features are better correlated with the manmade concepts in an image that such system tries to discover. Those principles could be used in general feature selection and extraction algorithms as well. This thesis showed so for microarray gene expression data sets.

We concentrated on the two class sample classification problems. Data sets that we dealt

with in this thesis always have binary class labels (cancer v.s. normal). There also exists more complex data sets where multiple labels are recorded. One of the data set we used is actually of multiple sample class labels (the multi-class data set). Since our algorithms only deal with binary class labels in their current forms, only cancer/normal distinction is used. One interesting question though is whether it is easy to expand our algorithms to the general case of multiple class labels. We could convert a multiple labeled data set into several “one v.s. all” binary labeled data sets and solve the sample classification problem for each of those data sets. Classification results could later be fused using an ensemble classifier. This is the normally followed method as construction and analyzing of a binary classifier is much easier than that of a multi-class classifier.

Chapter 7

List of Publications

Xian Xu, Aidong Zhang, “**Virtual Gene: A Gene Selection Algorithm for Sample Classification on Microarray Datasets**”, *Computational Science C ICCS 2005: 5th International Conference, 2005 International Workshop on Bioinformatics Research and Applications*, Atlanta GA, USA, 2005.

Xian Xu, Aidong Zhang, “**Virtual Gene: Using Correlations Between Genes to Select Informative Genes on Microarray Datasets**”, *LNCS Transactions on Computational Systems Biology II, LNBI 3680*, 138-152, 2005.

Xian Xu, Aidong Zhang, “**Selecting Informative Genes from Microarray Dataset by Incorporating Gene Ontology**”, *In Proceedings of IEEE BIBE 2005*, Minneapolis MN,

USA, 2005.

Xian Xu, Aidong Zhang, “**Improving Feature Subset Selection By Boosting: An Empirical Study On Microarray Data Set**”, *Computational Science C ICCS 2006: 6th International Conference, 2006 International Workshop on Bioinformatics Research and Applications*, Reading UK, 2006.

Xian Xu, Aidong Zhang, “**Improving Feature Subset Selection By Boosting: An Empirical Study On Microarray Expression Data Set**”, *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, to appear, 2006.

Xian Xu, Aidong Zhang, “**Feature Extraction By Incorporating Gene Ontology Annotations For Microarray Expression Data Set**”, *Bioinformatics*, submitted, 2006.

Bibliography

- [1] bioinformatics.org, <http://bioinformatics.org/faq/#definitions>.
- [2] Gene-chips, <http://www.gene-chips.com/>.
- [3] http://en.wikipedia.org/wiki/amino_acid.
- [4] <http://medstat.med.utah.edu/block2/biochem/formosa/menu.html>.
- [5] <http://www.ws.binghamton.edu/fridrich/562/fld.pdf>.
- [6] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *in Proc. of ACM SIGMOD 1993*, 1993.
- [7] Ftima Al-Shahrour, Ramn Daz-Uriarte, and Joaquin Dopazo. Fatigo: a web tool for finding significant associations of gene ontology terms with groups of genes. *Bioinformatics*, 20(4):578–580, 2004.

- [8] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissue probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. U.S.A.*, 96(12):6745–50, 1999.
- [9] Amigo. Amigo, <http://www.genedb.org/amigo/perl/go.cgi>.
- [10] Howard Anton. *Elementary Linear Algebra, 8th edition*. Wiley, 2000.
- [11] Francisco Azuaje and Olivier Bodenreider. Incorporating ontology-driven similarity knowledge into functional genomics: An exploratory study. In *In Proc. of IEEE BIBE 2004*, 2004.
- [12] 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(6):509–519, 2001.
- [13] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *In proceedings of the SIGMOD Conference 1990*, pages 322–331, 1990.
- [14] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini. Tissue classification with gene expression profiles. volume 7, pages 559–83, 2000.

- [15] T.H. Bø and I. Jonassen. New feature subset selection procedures for classification of expression profiles. *Genome Biology*, 3(4):research0017.1–0017.11, 2002.
- [16] G. V. Bobashev, S. Das, and A. Das. Experimental design for gene microarray experiments and differential expression analysis. *Methods of Microarray Data Analysis II*, pages 23–41, 2001.
- [17] Nadia Bolshakova, Francisco Azuaje, and Pdraig Cunningham. A knowledge-driven approach to cluster validity assessment. *Bioinformatics*, 21(10):2546–2547, 2005.
- [18] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, pages 144–152, 1992.
- [19] U. M. Braga-Neto and E. R. Dougherty. Is cross-validation valid for small-sample microarray classification? *BIOINFORMATICS*, 20(3):374–380, 2004.
- [20] Leo Breiman. Bagging predictors. *Machine Learning*, 1996.
- [21] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–77, 1973.
- [22] Debouck C. and Goodfellow PN. Dna microarrays in drug discovery and development. *Nature Genetics*, 21(1 suppl):48–50, 1999.

- [23] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines.
- [24] Y. Chen, E.R. Dougherty, and M.L. Bittner. Ratio-based decisions and the quantitative analysis of cdna microarray images. *J. Biomed. Opt.*, 2:364–67, 1997.
- [25] J Cheng, S Sun, A Tracy, E Hubbell, J Morris, V Valmееkam, A Kimbrough, MS Cline, G Liu, R Shigeta, D Kulp, and MA. Siani-Rose. Netaffx gene ontology mining tool: a visual approach for microarray data analysis. *Bioinformatics*, 20(9):1462–1463, 2004.
- [26] G. Cong, K. H. Tung, X. Xu, F. Pan, and J. Yang. Farmer: finding interesting rule groups in microarray datasets. In *In Proc. of SIGMOD 2004*, 2004.
- [27] GO Consortium. Go consortium, <http://www.geneontology.org/>.
- [28] G.O. Consortium. The gene ontology (go) database and informatics resource. *Nucleic Acids Research*, 32:D258–D261, 2004.
- [29] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.
- [30] F. Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.
- [31] Xiangqin Cui and Gary A. Churchill. Statistical tests for differential expression in cdna microarray experiments. *Genome Biol.*, 4(4):210, 2003.

- [32] Maximilian Diehn, Gavin Sherlock, et al. Source: a unified genomic resource of functional annotations, ontologies, and gene expression data. *Nucleic Acids Research*, 31(1):219–223, 2003.
- [33] T. G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.
- [34] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *Proc. Computational Systems Bioinformatics*, page 523, 2003.
- [35] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.
- [36] 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(457):77–87, 2002.
- [37] I. Dunham, N. Shimizu, B.A. Roe, S. Chissoe, A.R. Hunt, J.E. Collins, R. Bruskiewich, D.M. Beare, M. Clamp, L.J. Smink, R. Ainscough, J.P. Almeida, A. Babbage, C. Bagguley, J. Bailey, K. Barlow, K.N. Bates, O. Beasley, C.P. Bird, S. Blakey, A.M. Bridgeman, D. Buck, J. Burgess, W.D. Burrill, and K.P. O'Brien. The dna sequence of human chromosome 22. *Nature*, 402(6761):489–95, 1999.
- [38] B. Efron, R. Tibshirani, J.D. Storey, and V. Tusher. Empirical bayes analysis of a microarray experiment. *J. Am. Stat. Assoc.*, 2001.

- [39] Bradley Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans*. SIAM, 1982.
- [40] Bradley Efron. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1994.
- [41] Bradley Efron. The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(467):619–42, 2004.
- [42] Bradley Efron and Robert Tibshirani. Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560, 1997.
- [43] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863C14868, 1998.
- [44] R.D. Fleischmann, M.D. Adams, O. White, R.A. Clayton, E.F. Kirkness, A.R. Kerlavage, C.J. Bult, J.F. Tomb, B.A. Dougherty, and J.M. Merrick. Whole-genome random sequencing and assembly of haemophilus influenzae. *Science*, 269(5223):496–512, 1995.
- [45] Flybase. Flybase, <http://flybase.bio.indiana.edu/>.
- [46] Roth F.P. Bringing out the best features of expression data. *Genome Res.*, 11:1801–1802, 2001.

- [47] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *in Proc. ICML 1996*, 1996.
- [48] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, (1):119–139, 1997.
- [49] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, (5):771–780, 1999.
- [50] 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. *Molecular Biology of the Cell*, 12(10):2987–3003, 2001.
- [51] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, , and P.O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11(12):4241–57, 2000.
- [52] Robert Gentleman, Vince Carey, Wolfgang Huber, Rafael Irizarry, and Sandrine Dudoit. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, 2005.
- [53] T. R. Golub et al. Molecular classifications of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–7, 1999.

- [54] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *In proceedings of the SIGMOD Conference 1984*, pages 47–57, 1984.
- [55] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.
- [56] M. A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, 1999.
- [57] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Pub, 2001.
- [58] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques 2nd edition*. Morgan Kaufmann, 2006.
- [59] T. Hastie, R. Tibshirani, M.B. Eisen, A. Alizadeh, R. Levy, L. Staudt, W.C. Chan, D. Botstein, and P. Brown. 'gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1(2), 2000.
- [60] M. Hattori, A. Fujiyama, T.D. Taylor, H. Watanabe, T. Yada, H.S. Park, A. Toyoda, K. Ishii, Y. Totoki, D.K. Choi, E. Soeda, M. Ohki, T. Takagi, Y. Sakaki, S. Taudien, K. Blechschmidt, A. Polley, U. Menzel, J. Delabar, K. Kumpf, R. Lehmann, D. Patterson, K. Reichwald, A. Rump, M. Schillhabel, and A. Schudy. The dna sequence of human chromosome 21. *Nature*, 405(6784):311–19, 2000.

- [61] Simon Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 1998.
- [62] M.A. Hearst, B. Scholkopf, S. Dumais, E. Osuna, and Platt. J. Trends and controversies - support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, 1998.
- [63] R. Herwig, A.J. Poustka, C. Muller, C. Bull, H. Lehrach, and J. OBrien. Large-scale clustering of cdna-fingerprinting data. *Genome Res.*, 9:1093–1105, 1999.
- [64] Desheng Huang and Wei Pan. Incorporating biological knowledge into distance-based clustering analysis of microarray gene expression data. *Bioinformatics*, 22(10):1259–68, 2006.
- [65] National Human Genome Research Institute. National human genome research institute: <http://www.accessexcellence.org/rc/vl/gg/microarray.html>.
- [66] I. Inza, B. Sierra, R. Blanco, and P. Larra naga. Gene selection by sequential search wrapper approaches in microarray cancer class prediction. *Journal of Intelligent and Fuzzy Systems*, 12(1):25–34, 2002.
- [67] J. Jaeger, R. Sengupta, and W. L. Ruzzo. Improved gene selection for classification of microarrays. In *Proc. PSB*, 2003.

- [68] A. K. Jain and D. Zongker. Feature selection: Evaluation application, and small sample performance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.
- [69] Anil K. Jain, Robert P.W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.
- [70] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, 2004.
- [71] Daxin Jiang, Jian Pei, Murali Ramanathan, Chun Tang, and Aidong Zhang. Mining coherent gene clusters from gene-sample-time microarray data. In *In proceedings of the KDD Conference 2004*, 2004.
- [72] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.
- [73] J. Khan, J.S. Wei, M. Ringner, L.H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C.R. Antonescu, and C. Peterson. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–9, 2001.
- [74] K. Kira and L. A. Rendell. A practical approach to feature selection. In *In Proc. of Machine Learning: Proceedings of International Conference, 1992 ICML*, 1992.

- [75] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.
- [76] I. Kononenko and E. Simec.
- [77] Thomas Lengauer. Bioinformatics: From the pre-genomic to the post-genomic era. *ERCIM News*, 2000.
- [78] Benjamin Lewin. *Genes VIII*. Prentice Hall; 1st edition, 2003.
- [79] T. Li, C. Zhang, and M. Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20:2429–2437, 2004.
- [80] H. Liu, J. Li, and L. Wong. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Inform.*, 13:51–60, 2002.
- [81] Y. Lu and J. Han. Cancer classification using gene expression data. *Genome Inform.*, 28:243–268, 2003.
- [82] Richard Maclin and David W. Opitz. An empirical evaluation of bagging and boosting. In *in Proc. AAAI-97*, 1997.
- [83] K. Mardia, J. Kent, and J. Bibby. *Multivariate Analysis*. Academic Press, 1979.

- [84] Geoffrey J. McLachlan, Kim-Anh Do, and Christophe Ambroise. *Analyzing Microarray Gene Expression Data*. Wiley, 2004.
- [85] Ron Mier and Gunnar Ratsch. An introduction to boosting and leveraging. *Advanced Lectures on Machine Learning, LNCS*, pages 119–184, 2003.
- [86] Malanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1998.
- [87] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [88] Annette M. Molinaro, Richard Simon, and Ruth M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–07, 2005.
- [89] S. Mukherjee and S. J. Roberts. A theoretical analysis of the selection of differentially expressed genes. *Bioinformatics*, 2004.
- [90] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9):917–922, 1977.
- [91] NCBI. National center for biotechnology information, <http://www.ncbi.nih.gov>.
- [92] D. Opitz. Feature selection for ensembles. In *in Proc. of 16th National Conf. on Artificial Intelligence, AAAI 1999*, pages 379–384, 1999.

- [93] W. Pan. A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. *Bioinformatics*, 18(4):546–554, 2002.
- [94] P. J. Park, M. Pagano, and M. Bonetti. A nonparametric scoring algorithm for identifying informative genes from microarray data. In *Proc. PSB*, 2001.
- [95] P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.
- [96] QuickGO. Quickgo, <http://www.ebi.ac.uk/ego/>.
- [97] S. Ramaswamy, P. Tamayo, R. Rifkin, S Mukherjee, C.H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J.P. Mesirov, T. Poggio, W. Gerald, M. Loda, E. S. Lander, and T.R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *PNAS*, 98(26):15149–15154, 2001.
- [98] D.F. Ransohoff. Rules of evidence for cancer molecular marker discovery and validation. *Nature Reviews/Cancer*, 4:309–13, 2004.
- [99] M. Robnik Sikonja and I. Kononenko. An adaptation of relief for attribute estimation in regression. In *In Proc. of Machine Learning: Proceedings of International Conference, 1997 ICML*, pages 296–304, 1997.

- [100] Marko Robnik-Sikonja and Igor Kononenko. Theoretical and empirical analysis of relief and rrelief. *Machine Learning Journal*, 53:23–69, 2003.
- [101] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
- [102] Alex Smola Klaus-Robert Müller Matthias Scholz Gunnar Rtsch Sebastian Mika, Bernhard Schlkopf. Kernel pca and de-noising in feature spaces. In *In proceedings of the NIPS Conference 1999*, 1999.
- [103] P. Sebastiani, E. Gussoni, I. S. Kohane, and M. Ramoni. Statistical challenges in functional genomics. *Statistical Science*, 18(1):33–70, 2003.
- [104] E. Segal, H. Wang, and D. Koller. Discovering molecular pathways from protein interaction and gene expression data. *BIOINFORMATICS*, 19(Suppl 1):264–272, 2003.
- [105] SGD. Sgd, <http://www.yeastgenome.org/>.
- [106] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge, 2004.
- [107] W. Siedlecki and J. Sklansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(2):197–220, 1988.

- [108] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, 10:335–347, 1989.
- [109] Chao Sima, Ulisses Braga-Neto, and Edward R. Dougherty. Superior feature-set ranking for small samples using bolstered error estimation. *Bioinformatics*, 21(7):1046–1054, 2005.
- [110] R. Simon, M.D. Radmacher, K. Dobbin, and L.M. McShane. Pitfalls in the use of dna microarray data for diagnostic and prognostic classification. *Journal of the National Cancer Institute*, 95(1):14–18, 2003.
- [111] D. K. Slonim. From patterns to pathways: gene expression data analysis comes of age. *Nature Genetics*, 32:502–8, 2002.
- [112] 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(12):3273–3297, 1998.
- [113] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, and T.R. Golub. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *PNAS*, 96:2907–2912, 1999.

- [114] C. Tang and A. Zhang. Mining multiple phenotype structures underlying gene expression profiles. In *Proceedings of the 2003 CIKM Conference*, 2003.
- [115] P.E. Hart T.M. Cover. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, (1):21–27, 1967.
- [116] 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(11):1454–61, 2002.
- [117] Johannes Tuikkala, Laura Elo, Olli S. Nevalainen, and Tero Aittokallio. Improving missing value estimation in microarray data with gene ontology. *Bioinformatics*, 22(5):566–72, 2006.
- [118] Virginia Goss Tusher, Robert Tibshirani, and Gilbert Chu. Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98(9):5116–5121, April 2001.
- [119] U.M. and E.R. Dougherty. Bolstered error estimation. *Pattern Recognition*, 37:1267–1281, 2004.
- [120] U.M. and E.R. Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20:374–380, 2004.
- [121] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

- [122] Andy Vierstraete. <http://users.ugent.be/~avierstr>.
- [123] H. Wang, W. Wang, J. Yang, and P.S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, 2002.
- [124] Haiying Wang and Francisco Azuaje. Gene expression correlation and gene ontology-based similarity: An assessment of quantitative relationships. In *In Proc. of IEEE CIBCB 2004*, 2004.
- [125] Yuhang Wang and Fillia Makedon. Application of relief-f feature filtering algorithm to selecting informative genes for cancer classification using microarray data. In *Proceedings of the 2004 CSB Conference*, 2004.
- [126] Yuhang Wang, Fillia S. Makedon, James C. Ford, and Justin Pearlman. Hykgene: a hybrid approach for selecting marker genes for phenotype classification using microarray gene expression data. *Bioinformatics*, 21(8):1530–1537, 2005.
- [127] Andrew Webb. *Statistical Pattern Recognition*. Newnes, 1999.
- [128] Andrew Webb. *Statistical Analysis of Gene Expression Microarray Data*. Chapman & Hall/CRC, 2003.

- [129] Y. Wu and A. Zhang. Feature selection for classifying high-dimensional numerical data. In *IEEE Conference on Computer Vision and Pattern Recognition 2004*, volume 2, pages 251–258, 2004.
- [130] E. P. Xing, M. I. Jordan, and R. M. Karp. Feature selection for high-dimensional genomic microarray data. In *Proc. 18th International Conf. on Machine Learning*, pages 601–608. Morgan Kaufmann, San Francisco, CA, 2001.
- [131] M. Xiong, L. Jin, W. Li, and E. Boerwinkle. Computational methods for gene expression-based tumor classification. *BioTechniques*, 29(6):1264–1270, 2000.
- [132] Xian Xu and Aidong Zhang. Selecting informative genes from microarray dataset by incorporating gene ontology. In *In Proceedings of IEEE BIBE 2005*, 2005.
- [133] Xian Xu and Aidong Zhang. Virtual gene: A gene selection algorithm for sample classification on microarray datasets. In *Computational Science ICCS 2005: 5th International Conference, 2005 International Workshop on Bioinformatics Research and Applications*. Springer-Verlag GmbH, 2005.
- [134] Xian Xu and Aidong Zhang. Virtual gene: Using correlations between genes to select informative genes on microarray datasets. *LNCS Transactions on Computational Systems Biology II, LNBI 3680*, pages 138–152, 2005.
- [135] Xian Xu and Aidong Zhang. Improving feature subset selection by boosting: An empirical study on microarray data set. In *Computational Science ICCS 2006: 6th*

International Conference, 2006 International Workshop on Bioinformatics Research and Applications. Springer-Verlag GmbH, 2006.

- [136] Xian Xu and Aidong Zhang. Improving feature subset selection by boosting: An empirical study on microarray expression data set. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 2006 to appear.
- [137] S. Ben Yahia, T. Hamrouni, and E. Mephu Nguifo. Frequent closed itemset based algorithms: A thorough structural and analytical survey. *SIGKDD Explorations*, 8(1):93–104, 2006.
- [138] L. Yu and H. Liu. Redundancy based feature selection for microarray data. In *Proc. of SIGKDD*, 2004.