Enhanced Machine Learning Techniques for Microarray Cancer Classification

by

SHEN LI

Submitted in fulfillment of the requirement for the degree of

Doctor of Philosophy

at

School of Computer Engineering

Nanyang Technological University

October 2005

ABSTRACT

This thesis deals with the problem of microarray cancer classification. Machine learning methods have been used and enhanced to tackle the task. Some novel use of the current methods has been presented. Evaluation based on classification accuracy and execution time has shown the advantages of the enhancements.

Dimension-reduction methods have been combined with logistic regression to solve the binary classification problems. Competitive performance and efficient calculation are observed comparing with the existing methods. In addition, the logistic regression method produces the value of probability directly so that the prediction strength is easier to interpret. Two dimension-reduction methods, namely, the principal component analysis (PCA) and the partial least squares (PLS), are used and compared with each other. It has been shown that the PLS is more superior to the PCA in various aspects. The PCA and PLS are further extended to kernel forms for solving nonlinear problems. They are then integrated with a regularized least-squares classifier (RLSC) for binary cancer classification. Nonlinear Gaussian kernel and linear kernel are both used. When comparing the two regularized classifiers, the RLSC and the support vector machine (SVM), the RLSC has shown its advantage in classification performance as well as in execution. The problem of component selection for PCA and PLS has been studied empirically and the proposed method has shown its effectiveness.

Multiclass cancer classification is dealt with in such a way that they are firstly decomposed into a few binary sub-problems and then combined according to a coding matrix. There are several ways for the decomposition and combination. Therefore, evaluations based on classification accuracy have been used to determine which ways are more preferred. Comparisons with other widely used multiclass methods such as

K-nearest neighbors, neural networks and C4.5 decision trees, have shown the superiority of the output-coding method. For the output-coding method, error-correcting output codes (ECOC) have been found to be superior. A novel method has been proposed to enhance the performance of the ECOC output-coding.

The problem of finding an optimal coding matrix is known to be NP-hard and cannot be solved in a brute force way. Genetic algorithm (GA) has been used to solve this problem by considering each coding matrix as a bitstring, or a chromosome in terms of evolution. It finds a coding matrix which is minimum in its fitness value by imitating the mechanism of natural selection. A novel criterion to evaluate the fitness of a coding matrix has been proposed. It is based on the upper bound of a training error of a coding matrix and is called a multiple margin to reflex the idea that each cancer class has a corresponding margin to be maximized. It has been shown that there is improvement in classification performance by using the multiple margin criteria. Some new insights have also been described about the relationship between the codeword length and the classification accuracy.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Tan Eng Chong, Associate Professor of School of Computer Engineering, Nanyang Technological University, who opens the gate of scientific research for me. He gives me enormous guidance and help and offers me many chances to work and think independently.

I would like to thank Nanyang Technological University for the financial support during the PhD study.

I would like to thank my parents, Shen Erguang and Wang Xinhua, for their continuous love and support. Wherever I go, the family support always makes me strong and secured.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

Acronym	Meaning
ALL	Acute Lymphoblastic Leukemia
AML	Acute Myeloid Leukemia
ANN	Artificial Neural Network
AP	All-Pairs
BREAST	Breast Cancer
BW	Between-Within Variances
C45	C4.5 decision tree
CNS	Central Nervous Systems Tumor
COLON	Colon Tumor
CPU	Central Processing Unit
CV	Cross-Validation
DIV	Diversity Measure
DNA	Deoxyribonucleic Acid
ECOC	Error-Correcting Output-Codes
FDA	Fisher Discriminant Analysis
FLDA	Fisher’s Linear Discriminant Analysis
GA	Genetic Algorithm
HAM	minimum hamming-distance
HAMD	Hamming-Distance Decoding
INNR	Inner-Product Decoding
KNN	K-nearest neighbors
KPCA	Kernel Principal Component Analysis
KPLS	Kernel Partial Least Squares
LD	Logistic Discrimination
LDA	Linear Discriminant Analysis
LEUKEMIA	Acute Leukemia
LOOCV	Leave-One-Out Cross-Validation
LOSS	Loss-based Decoding
LS-SVM	Least Squares SVM

LSR	Least Squares Regression
LUNG	Lung Cancer
MC-SVM	Multi-Category SVM
MLM	Multiple Margins
MLP	Multi-layer Perceptron
MS	Mass Spectrometry
MSE	Mean-Squared-Error
NN	Neural Network
NO	No Feature Selection
NP	Non-Polynomial (exponential)
OVA	One-Versus-All
OVARIAN	Ovarian Cancer
PCA	Principal Component Analysis
PLR	Penalized Logistic Regression
PLS	Partial Least Squares
PPLS	Penalized PLS
PROB	Probabilistic Decoding
PROSTATE	Prostate Cancer
QDA	Quadratic Discriminant Analysis
QP	Quadratic Programming
RAN	Pure Random
RBF	Radial Basis Function
RFE	Recursive Feature Elimination
RHC	Randomized Hill-Climbing
RKHS	Reproducing Kernel Hilbert Space
RLSC	Regularized Least Squares Classification
RNA	Ribonucleic Acid
SVD	Singular Value Decomposition
SVM	Support Vector Machine
VOI	Variance of Input Variables
VOL	Variance of Label
WV	Weighted Voting

List of Symbols

Symbol	Meaning
$\leftarrow$	Updating original variable
$\mathbf{1}_m$	$m \times 1$ vector with all entries equal to one
α	Logistic regression parameter for bias
$\tilde{\alpha}$	Estimate of α
β_j	Logistic regression parameter for j th variable
$\tilde{\beta}_j$	Estimate of β_j
$\boldsymbol{\beta}$	$[\beta_1, \beta_2, \ldots \beta_n]^T$
$\tilde{\boldsymbol{\beta}}$	Estimates of $\boldsymbol{\beta}$
$\bar{\boldsymbol{\beta}}$	Averaged $\boldsymbol{\beta}$
ε	Average binary loss
ξ	Slack variable in SVM
η	Logistic link function (Chapter 2)
η	Class Margin (Chapter 5)
η_i	Linear predictor for the i th sample (Chapter 2)
η_i	Margin for the i th class (Chapter 5)
$\tilde{\eta}_i$	$\tilde{\alpha} + \sum_{j=1}^{n} \tilde{\beta}_j x_{ij}$
$\tilde{\boldsymbol{\eta}}$	$[\tilde{\eta}_1, \tilde{\eta}_2, \ldots \tilde{\eta}_m]^T$
H	Reproducing kernel Hilbert space (RKHS)

$\boldsymbol{\theta}$	Regression coefficients for components
$\tilde{\boldsymbol{\theta}}$	Estimates of $\boldsymbol{\theta}$
λ	Penalty parameter
μ	Mean value
ρ	Minimum Hamming-distance between all rows
σ	Standard deviation
$\bar{\sigma}$	Standard deviation of μ
Φ	Mapping function from input space to feature space
$\boldsymbol{\Phi}$	Mapping of $\mathbf{X}$ in RKHS
ω	Width parameter for Gaussian kernel
b	Unregularized bias
$\mathbf{b}$	A $m \times 1$ vector with all entries being b
C	Regularization parameter for SVM
C_{ij}	Column separation between column i, j
d	Decoding function
d_H	Hamming-distance decoding function
d_I	Inner-product decoding function
d_L	Loss-based decoding function
d_P	Probabilistic decoding function
D	Fitness value using diversity measure
D_c	Normalized average column separation
D_r	Normalized average row separation
f	Function for a classifier

f_s	s th binary classification function
$\mathbf{f}$	$(f_1, f_2, \dots f_l)$
H	Fitness value using minimum Hamming-distance measure
H_c	Normalized minimum column separation
H_r	Normalized minimum row separation
I	Indicator function
$\mathbf{I}$	Identity matrix
J	Cost function
J^*	Cost function for LSR
k	Number of components (Chapter 2, 3)
k	Number of classes (Chapter 4, 5)
$\mathrm{K}_{\mathrm{ker}}$	Kernel function
$\mathbf{K}$	Kernel matrix
$\mathbf{K}_t$	Test set kernel matrix
$\mathbf{K}_k$	Remaining kernel matrix after k iterations
l	Codeword length
l_i	i th element of score vector $\mathbf{l}$
$\mathbf{l}$	Score vector for $\mathbf{y}$
$\mathbf{L}$	Score vectors for $\mathbf{y}$
L	Log-likelihood
L^*	Penalized log-likelihood
$L(\cdot,\cdot)$	Loss function
m	Number of samples

m_k	Number of samples in class k
m_t	Number of testing samples
M	Fitness value using multiple margin measure
M_r	Averaged margin values for all classes
$\mathbf{M}$	Coding matrix
n	Number of variables
n_0	Dimension of RKHS
p	Probability that a sample is class 1
p_i	The probability that $y_i = 1$
$\tilde{p}_i$	Estimate of p_i
p_s	Probabilistic output of binary classifier s
$\mathbf{p}$	$[p_1, p_2, \dots, p_m]^T$
r_l	Proportion of VOL captured by k components
r_v	Proportion of VOI captured by k components
r_i	i th bit of codeword
$\mathbf{r}$	Codeword
$\mathbf{r}_i$	Codeword for the i th class
R_{ij}	Hamming-distance between row i, j
$\mathbf{R}$	Residual matrix
$\Re$	Input space
s_i	i th singular value
$\mathbf{s}_j$	Column represents the j th binary classifier
$\mathbf{S}$	Singular value diagonal matrix

t_i	i th element of score vector $\mathbf{t}$
t_{ij}	j th element of i th score vector
$\mathbf{t}$	Score vector for $\mathbf{X}$
$\mathbf{t}_i$	i th score vector
$\mathbf{T}$	Score vectors for $\mathbf{X}$
$\mathbf{T}_t$	Test set score vectors
$\mathbf{u}$	a m -vector of ones
$\mathbf{U}$	Left singular vectors
$\mathrm{V}(\cdot,\cdot)$	Loss function
$\mathbf{V}$	Loading vectors
w_i	i th element of weight vector
$\mathbf{w}$	Weight vector
x_j	The j th gene expression level of a sample
x_{ij}	The j th gene expression level of i th sample
$\mathbf{x}$	$[x_1, x_2, \ldots, x_n]^T$
$\mathbf{x}_i$	The i th sample
$\mathbf{X}$	a $m \times n$ matrix so that $\mathbf{X}(i,j) = x_{ij}$
y	Sample class
y_i	The class of the i th sample
$\bar{y}$	$\sum_{i=1}^{m} y_i \big/ m$
$\mathbf{y}$	$[y_1, y_2, \ldots, y_m]^T$
$\mathbf{z}_i$	The i th testing sample

Chapter 1. Introduction

1.1 Current Development

Many different biological questions are routinely studied using transcriptional profiling on microarrays (Schena *et al.*, 1996; DeRisi *et al.*, 1997; Wodicka *et al.*, 1997; Chu *et al.*, 1998). A wide range of approaches is available for gleaning insights from the data obtained from such experiments.

The widespread application of DNA microarrays to cancer research is astounding. In the short ten-year history of this versatile technology, hundreds of large-scale experiments have been done, generating global quantitative profiles of gene expression in cancer (Schena *et al.*, 1995; Schena *et al.*, 1996; Alizadeh *et al.*, 2000; West *et al.*, 2001). Along with the rapid development of microarray technologies, there has been an extraordinary amassment of data. The appropriate choice of data-analysis technique depends both on the data and on the goals of the experiment.

Storage and analysis of these data can be a pain for microarray researchers. Although at present there is no clear standard solution for microarray data storage and analysis software, there are many open-source, public domain and commercial solutions rivaling for a share of this evolving market. Most of the available products are still in the early phases of the software development process; consequently, new and improved versions of these softwares are being released frequently to keep up with consumer expectations and to provide “patches”. An exhaustive record of microarray software can be found on Y.F. Leung’s website: http://ihome.cuhk.edu.hk/～b400559/array.html.

This thesis presents some enhancements to the data-analysis methods that are used to deal with an important task: microarray cancer classification.

1.2 Microarray Technology

It is known that nucleotide-acids stored in chromosomes carry the information of all living things. The instructions to perform various cellular activities are stored in the form of A, C, G and T to denote four different types of bases under normal conditions (Weaver, 2004). One gene may consist of thousands of such bases while one DNA strand may consist of hundreds of genes. All DNAs in cells of different tissues are the same. However, these tissues look quite different. The Central Dogma (Weaver, 2004) of molecular biology tells us that DNA has to be transcripted into messenger-RNA (mRNA) and then translated into proteins to perform its embedded program. Almost all cellular functions are carried out by proteins and they are the building blocks of cell structures. Therefore, the mRNA and protein levels depict the status of the cells of a particular tissue.

There are two types of DNA microarrays (Lockhart and Winzeler, 2000), namely, cDNA and oligonucleotide. In cDNA microarrays, the ratios of two kinds of gene expression levels are obtained. Genes of the samples of interests are called targets while genes of the control samples are called probes. The targets and probes are hybridized and spotted on a solid glass plate so that the fluorescence images corresponding to the relative gene expression abundance can be scanned (Figure 1.1a). The numbers of gene expression abundance can then be obtained using some image analysis software tools.

In oligonucleotide microarrays, the absolute values of gene expression levels are obtained. For each gene, multiple probes of 20-mers are synthesized base by base using photolithography in a particular position on a glass plate. The quantitative fluorescence images of microarrays are scanned and processed thereafter (Figure 1.1b).

Another technique called proteomic mass spectrometry (MS) can be used to screen the protein levels in a complex mixture (e.g. human blood serum). MS is often used to test for the presence or absence of one or more molecules (Lilien *et al.*, 2003). The MS data therefore can also be used for cancer classification. Although they are not microarray data, there are some similarities between MS and microarray data. Both of the two types of data have large-dimension (thousands) of variables along with relatively small size (tens or hundreds) of samples because of the high cost of experiments. Therefore, the two types of data are treated equally in this thesis. There is one MS dataset used for cancer classification in later chapters.

1.3 Machine Learning and Molecular Cancer Classification

Advances in microarray technology have made it possible to address the problem of whether the gene expression profiles can be used to classify cancerous

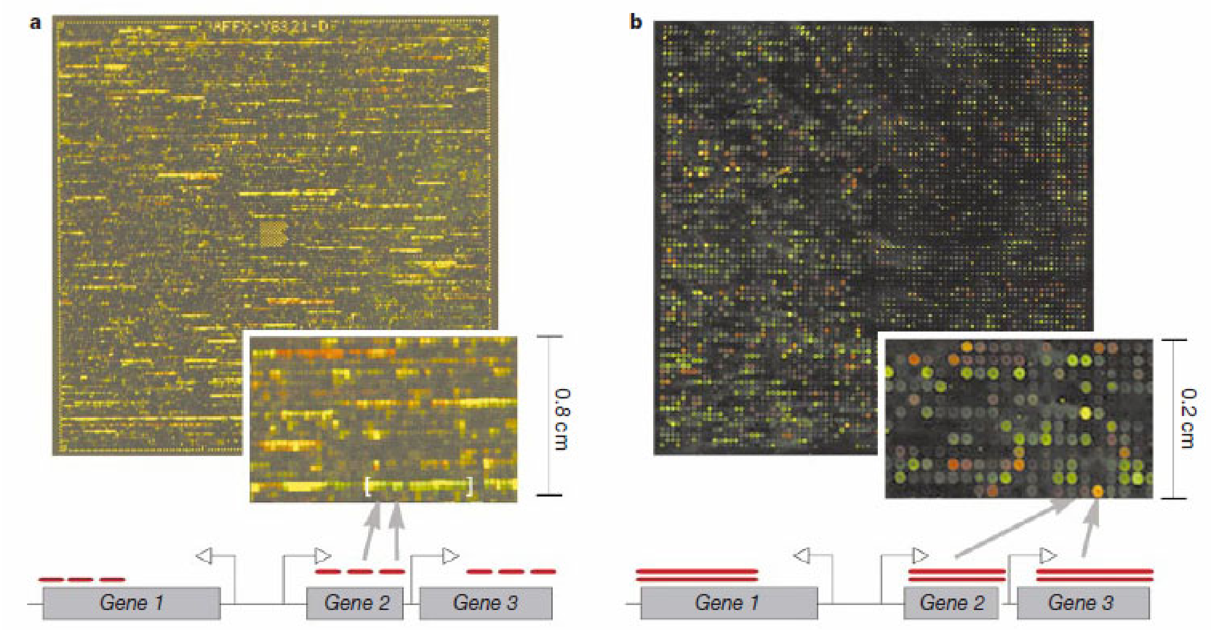


Figure 1.1 Microarrays. (a) Oligonucleotide; (b) cDNA. (Lockhart and Winzeler, 2000)

samples (or different subtypes of certain cancers) more accurately and efficiently (Alon *et al.*, 1999; Golub *et al.*, 1999). It is also important to address the problem called feature selection in the context of microarray data so that a very small subset of genes that strongly relate to diseases can be selected as genetic fingerprints to further reduce the independent variable domain.

This type of problems belongs to the category of supervised learning. Given a training dataset, classifiers are built from the data in which the variable values and the class labels of samples are known *apriori*. The learnt models are then verified using samples that are hidden during training process. It has already been shown that very low error rates can be achieved by using machine learning methods (Alon *et al.*, 1999; Golub *et al.*, 1999). The field known as microarray cancer classification has received intensive research efforts ever since. Many methods originated from machine learning have been adapted and successfully applied to cancer classification (Ben-Dor *et al.*, 2000; Furey *et al.*, 2000; Khan *et al.*, 2001; Dudoit *et al.*, 2002; Guyon *et al.*, 2002; Shen and Tan, 2005a). Several of them will be introduced in later chapters.

Machine learning consists of several subfields like statistical learning, neural networks, genetic algorithm and decision tree learning. This thesis mainly emphasizes on statistical learning while briefly describes the usages of other methods.

1.4 Existing Challenges

Microarray experiments produce thousands of gene expression levels simultaneously but suffer from high cost at present. Therefore, the size of samples is usually in the order of tens or hundreds. Some traditional regression methods cannot be directly applied because of the inversion of a singular matrix. There is another "curse of dimensionality" problem which causes some classifiers unable to perform well on testing data. These problems will be investigated in later chapters. Dimension reduction is also an important task since the large gene set is always intended to be reduced. By using only a few components constructed from the original data, some structures of the gene expression profiles may be revealed through visualization.

Multiclass classification is much more difficult than binary one and remains a challenge in machine learning research. Accurate prediction of multiple class labels for microarray samples is an ongoing research task.

1.5 Objectives

Both binary and multiclass cancer classification problems will be investigated. For binary classification, dimension reduction will be combined with some traditional regression methods to see whether it is more efficient to solve the singular matrix inversion problem. This may also enhance performance as well as execution. Several issues regarding the application of the new methods will be studied. Systematic comparisons regarding classification accuracies and computational time with other methods will be made. Feature selection will be used to reduce the gene subset from thousands to hundreds or even less than ten without losing classification performance. Combination of kernelized dimension reduction with traditional regression methods will also be investigated. Comparisons with other kernel classifiers will be made to see how the new methods are superior in performance.

For multiclass classification, the application of output-coding with binary classifiers is to be investigated. Comparisons with other multiclass classification methods will be made to show the performance gain by the output-coding methods. Genetic algorithm (GA) will be employed to see whether it can enhance the classification performance by optimizing coding-matrices. Because codeword length is an important part of coding-matrix design, the relationship between codeword length and multiclass classification performance will also be investigated.

1.6 Organization of the Thesis

Chapter 1 gives an introduction of the current development of microarray cancer classification and describes the existing challenges and the objectives of this thesis.

Chapter 2 proposes to combine dimension reduction with the logistic regression to solve the singular matrix inversion problem. The new method is compared with some of the state-of-the-arts classifiers in both performance and computational cost. Two different dimension reduction methods, namely partial least squares and principal component analysis, are compared in various aspects.

Chapter 3 extends the dimension reduction methods to kernel form. A traditional regression method, regularized least-squares classification, is integrated with the kernel dimension reduction methods. Experiments are given to show its performance and efficient execution.

Chapter 4 proposes to use output-coding to solve the multiclass classification problem. Several aspects like coding strategy, decoding function and feature selection are discussed. Comparison with other multiclass methods like decision trees, neural networks and nearest neighbors is performed.

Chapter 5 proposes to use GA to solve the problem of jointly maximizing the row separation and the column separation of a coding matrix. A new criterion is proposed to evaluate the "fitness" of a coding matrix. Comparisons are made to show the advantage of the new criterion and the effectiveness of genetic algorithm in solving the optimization problem. The relationship between the codeword length and the classification performance is carried out and some new insights are given.

Chapter 6 concludes the work that was conducted in the course of this research and discusses future works.

Chapter 2. Dimension Reduction Based Penalized Logistic Regression for Cancer Classification

2.1 Introduction

DNA microarrays are capable of profiling gene expression patterns of tens of thousands of genes in a single experiment. DNA targets are arrayed onto glass slides (or membranes) and explored with fluorescent- or radioactivly-labelled probes (Brown and Botstein, 1999; Debouck and Goodfellow, 1999; Duggan *et al.*, 1999). Wealth of this kind of data in different stages of cell cycles helps to explore gene interactions and to discover gene functions. Moreover, obtaining genome-wide expression data from cancerous tissues gives insight into gene expression variation of various tumor types, thus providing clues for cancer classification of individual samples. Accurate diagnosis of tumor types can enhance efficacy and reduce toxicity of medical treatment for cancer patients. During the past decades, cancer classification has been relying on subjective judgment from experienced pathologists. Microarray experiments can be employed to screen gene expression levels from cancerous and normal tissue samples so that proper prediction rules can be built from these gene expression data.

Machine learning techniques have been successfully applied to cancer classification using microarray data (Peterson and Ringnër, 2003). Among these techniques, the supervised learning methods are most popular for cancer diagnosis. Given a set of training samples, the gene expression levels and class label of each sample are known. The goal of supervised learning method is to build a prediction rule by properly setting up the parameters of a classifier with these training data. After that, the classifier should be able to assign correct labels to new microarray samples that come along. However, microarray data have the characteristics that the number of samples is much less than the number of variables and this causes the "curse of dimensionality" problem. This may cause problems for estimating parameters properly; for instance, data overfitting may always occur. Some state-of-the-arts methods can handle this situation; for example, the support vector machines (SVM) (Furey *et al.*, 2000) have been found to be effective and robust for cancer classification using microarray data.

Although cancer classification has been intensively researched for years and very high accuracy has been achieved by dozens of machine learning methods, most of these techniques lack the following properties: First, the prediction strength cannot be explicitly given. Sometimes, the probability of an output itself is the thing of interest and an incorrect diagnosis may cause severe consequence. Second, there is a lack of interpretation of the weights or the regression coefficients of predictor variables. The contribution of each predictor variable for the output of prediction may help biologists finally discover the genes that interact and cause the occurrences of diseases.

The penalized logistic regression (PLR) technique (Eilers *et al.*, 2001; Schimek, 2003; Zhu and Hastie, 2004) has intrigued us to tackle the above-mentioned problems. Some researchers have applied the method to cancer classification and found that it performs as well as the SVM. As an investigation, we combined the dimension-reduction methods, singular value decomposition (SVD) and partial least squares (PLS), respectively with the PLR in order to improve both the training speed and the classification accuracy. We found that PLR+PLS is preferred to PLR+SVD and the former performs as well as or even better than other available methods. Several issues relating to the application of our new methods, such as components selection and the choice of penalty parameters are discussed.

Feature selection is another important issue in cancer classification. By removing features that are irrelevant to cancers, the accuracy of prediction can usually be improved. With a subset of much fewer genes, the contribution of each gene will be more prominent and even the interaction of these genes can be revealed by using other techniques such as the Bayesian networks. We used a method called recursive feature elimination (RFE) to iteratively select a series of nested gene subsets and found that some of these gene subsets could achieve almost perfect classification with much less genes than the original set of genes.

2.2 Methods

2.2.1 A Brief Overview of Machine Learning for Cancer Classification

Classification problem has been extensively studied in the area of statistics, machine learning and databases. In the past few years, researchers have started paying attention to cancer classification using gene expression. Most proposed cancer classification methods are from the statistical and machine learning area. Machine learning is closely related with statistics but it emphasizes on the algorithmic complexity of computational implementation.

The main problems that we concern in cancer classification are accuracy and computational time. There is no single classifier that has been proved to be superior over the rest. In the following, several commonly used classification methods are briefly reviewed.

1. *Weighted voting (WV) of informative genes*. The weighted voting method is proposed by Golub *et al*. (1999) for classifying the binary-class data. In the WV method, the assignment of classes is based on the weighted voting of the expression values of a group of "informative genes" in the test samples. The informative genes are genes that have high correlation with

the class labels. A "prediction strength" (PS) is given by the sum of the correlations of all informative genes in a test sample. A "prediction strength threshold" (PST) is used to determine if the prediction of the weighted voting is strong enough to assign the majority class to the test sample. If $\mathrm{PS} \geq \mathrm{PST}$, then the winning class is assigned to be the class label of the test sample; otherwise, the weighted voting is considered to be "Uncertain" as the class label to the test sample. The WV method is simple and works well in some data such as the Leukemia data set. But its simplicity also results in some limitations. First, it is only applicable for binary-class problem. Cancer classification would involve identifying of more than two classes of cancer. In this case, WV algorithm will not be effective. Second, it chooses an equal number of genes that have high correlation with the two classes. This means that it is only effective in classifying data sets that are unbiased. If the majority genes of a data set are highly correlated with one class, then this method will not produce a satisfactory classification result.

2. *Artificial Neural networks*. ANNs are computer-based algorithms that emulate the structure and behavior of neurons in the human brain. They can be trained to recognize and classify complex patterns. Pattern recognition is achieved by adjusting parameters of the ANN by a procedure of error minimization through learning from data. They can be applied to any type of input data, such as gene-expression levels generated by cDNA microarrays, and the output can be grouped into any given number of classes. Khan *et al*. (2001) used ANNs for cancer classification. They also used PCA as dimension reduction before applying ANNs for

classification. Neural network method has comparable result with the other methods. But it does the classification in a black box manner. The users are not provided with any information on how the genes are correlated, which set of genes is more effective for classification, etc.

3. *Decision tree*. A decision tree consists of a set of internal nodes and leaf nodes. The internal nodes are associated with a splitting function which consists of a splitting attribute and one or more splitting predictors defined on this attribute. Each leaf node is labeled with a single class label. The construction of the decision tree is usually a two phase process. In the first phase, the growing phase, an overgrown decision tree is built from the training data. The splitting function at each internal node is chosen to split the data sets into subsets that have better class separability, thus minimizing the misclassification error. In the second phase, the pruning phase, the tree is pruned using some ad hoc methods to avoid overfitting of data. However, this process tends to introduce classification errors on the test data. Zhang *et al*. (2001) proposed a recursive partitioning cancer classification method based on a binary decision tree. Decision trees have been attractive in pattern classification environment for several reasons. First, their results are interpretable. Second, they do not require any parameter input. Third, the construction process is relatively fast. These advantages are also applicable to cancer classification. An additional advantage is the scalability of decision trees.

4. *Fisher's linear discriminant analysis* (FLDA). FLDA is a non-parametric method that finds a projection matrix P which reshapes the data set to maximize the class separability. Class separability is defined to be the ratio

of the between-class scatter matrix to the within-class scatter matrix. This projection defines features that are optimally discriminating. Dudoit *et al.* (2002) applied FLDA to cancer classification problem and they reported that it had the highest error rates compared with the rest of the classifiers that they studied. They reasoned that, given a limited number of samples and a very large number of genes, the between-group and within-group scatter matrices are quite unstable and provide poor estimates of the corresponding population quantities.

5. *K-Nearest Neighbor (KNN)*. This method is based on a distance metric between the testing samples and the training samples. The main idea of the method is, for each testing sample s; find k training samples with most similar expression patterns, according to a distance measure. The class label of s is then determined by voting of the k training samples. The distance metric can be any similarity measure based on attribute values, for example, the Pearson correlation function, the Euclidean distance function, etc. Dudoit *et al.* (2002) applied KNN to several types of tumor classification problems. Since KNN bases its classification on the similarity of the expression values of each gene, this makes it less prone to noise and bias in the data. The disadvantage of KNN is due to its non-scalability. Every testing sample needs to be checked against every training sample in order to find the most similar ones. This works fine when the data size is small, but it suffers when the data size is large.

6. *Aggregated classifiers: boosting*. Aggregated classifiers are built from the aggregation of multiple versions of the classifiers using majority voting. The idea is to generate multiple versions of classifiers over the training

data through bootstrapping. In each sampling, the training data are sampled with replacement by putting more emphasis on the training samples that were misclassified in the previous sampling. The final aggregated classifier is a weighted voting of the classifiers built during each sampling. The weight is dependent on the accuracy of the classifier built from the bootstrapped samples. Boosting is useful for improving the accuracy of the classifiers that are unstable to small changes of the learning set; for reducing the overfitting problem due to small training data set; for improving the accuracy of those weak classifiers. Ben-Dor *et al.* (2000) and Dudoit *et al.* (2002) applied boosting to several cancer classification problems. Boosting achieves comparable classification performance with respect to other methods. However, boosting is quite time-consuming because it is usually applied to weak learners to improve classification accuracy through repeated classification of the weighted training samples. Since the rest methods can also achieve comparable accuracy, boosting is not very attractive in cancer classification.

7. *Support vector machines (SVM)*. The SVM learning algorithm constructs a hyperplane with maximum margin that separates the positive samples from the negative samples. The margin of the hyperplane is defined as the distance from the hyperplane to the sets of points that are closest to it. The points that lie closest to this max-margin hyperplane are called the support vectors. The hyperplane can be defined using these points alone and the classifier only makes use of these support vectors to classify test samples. Sometimes when the different training classes are overlapped, the training samples may not be linearly separable in the feature space. In other times,

for the sake of error tolerance and overfitting avoidance, perfect linear separation may not be desirable. For these cases, it is better to allow some training samples to fall to the wrong side of the hyperplane. Soft-margin SVMs are developed to control the overall training errors through parameter tuning. SVMs are widely used for cancer classification purposes (Ben-Dor *et al*., 2000; Furey *et al*., 2000; Lee and Lee, 2003; Ramaswamy *et al*., 2001). The ability of SVMs for producing hyperplane with max-margin and for tuning the amount of training errors has made SVMs especially suitable for cancer classification.

SVM algorithm was developed in the 1990's. Since then, it has received intensive attentions and has been exploited by researchers from different fields, like data mining, pattern recognition and medical diagnosis. There is a sense that SVM outperforms a wide variety of other binary classification algorithms, such as ANN, decision trees, FLDA, KNN, and so forth. There is also a sense that SVM is based on a strong theoretical foundation, i.e., Vapnik's theory of VC-dimension. SVM's ability to deal with high dimensional data, to cope with overfitting, and robustness with noise and sparseness of solution all make it well suited for cancer classification. Another benefit offered by SVM is its scalability (Yu, 2004): the number of support vectors selected by the learning algorithm is usually small, even with a large training set. This is essential since the amount of available gene expression data will soon increase dramatically.

One disadvantage that limits SVM's applications is that it was originally developed for binary classification. Therefore, it is not straightforward to use it for multiclass problems. However, as will be shown in Chapters 4 & 5, that by combining with output-coding, SVMs can also be conveniently used for multiclass classification and can achieve very good performance.

SVM also provides the contribution of each attribute to a classifier through training. This is particularly interesting because feature selection, which is very important for cancer classification, can be based on the values of the classifier's coefficients. Studies have shown that by combining SVM and recursive feature elimination, better classification performance has been achieved than by using simple statistics as feature selection method that is separated from a classifier (Guyon *et al*., 2002).

SVM is closely related with regression methods from statistics. They share several analogous properties like the representation of a classifier, using regularization parameter to control data overfitting and providing prediction strength of a test sample. It will be shown in Chapter 3 that both SVM and regularized least squares classification (RLSC) can be derived from a single theoretical framework, i.e. Tikhonov regularization. And both of them can attain similar performance. In this chapter, another regression method, called PLR, will be emphasized and well studied.

2.2.2 Penalized Logistic Regression

Assume we have a number of cancer classification samples from microarray experiments. Each sample can be in one of the two classes: class 0 and class 1, say. A rule based on logistic regression is to be determined, which uses the gene expression profiles on an array to determine the probability that a sample belongs to one of the two classes. A training dataset of samples with known class labels is present to derive the rule and the rule derived should be able to classify any new sample that comes along.

Let a variable y indicate the class of a microarray sample: $y = 0$ means that the sample belongs to class 0; $y = 1$ means that the sample belongs to class 1. Let x_j indicate the j th gene expression level of the sample. The attempt is to find a formula that gives the probability p that the sample with all its measured expression $\mathbf{x}^T = [x_1, x_2, \ldots, x_n]$ represents a class 1 case. Since only two classes are considered, the probability of the sample representing class 0 is consequently $1 - p$. The normal logistic regression model would be

$$\eta = \log \frac{p}{1-p} = \alpha + \sum_{j=1}^{n} \beta_j x_j \qquad (2.1)$$

where α and $\beta_1, \beta_2, \ldots \beta_n$ are parameters which could be estimated by maximum likelihood (ML) criterion. The Newton-Raphson algorithm (Hastie *et al.*, 2001) can be used to solve this problem and the curve that computes p from η is given by

$$p = \frac{1}{1 + e^{-\eta}} \qquad (2.2)$$

which is called the logistic curve, hence the name logistic regression.

In the setting of microarray experiments, the number of samples, m, is usually on the order of tens or hundreds but the number of variables, n, is usually on the order of thousands or even tens of thousands. So the number of samples is much less than the number of variables. There are three problems in this situation when we attempt to build a logistic regression rule:

1. If $m < n$, there will be more unknowns than equations; possible solutions are infinite.
2. Data overfitting may occur. That means we may have zero errors on training data but very poor performance on new samples.

3. Multi-collinearity largely exists. Many genes will show nearly identical patterns across the samples, so they supply no new information on the data; some gene profiles can be linear combinations of the other gene profiles.

These problems can be solved by introducing a penalty into the logistic regression formulation. The penalty on the sum of the squares of the regression coefficients is known as ridge regression (Hoerl and Kennard, 1970). It has been applied to logistic regression by (le Cessie and van Houwelingen, 1992). Dimension-reduction methods such as the PLS and the SVD can be used to tackle the above problems and make the computation of PLR feasible. The efficiency of classifiers can be enhanced by using the dimension-reduction methods as well. The PLS and SVD will be introduced in the next sub-section. The PLR is described in the following.

Let y_i indicate the class of the ith sample and p_i the probability that $y_i = 1$. Let x_{ij} indicate the jth gene expression level of the ith sample. Then the model is

$$\eta_i = \log \frac{p_i}{1-p_i} = \alpha + \sum_{j=1}^{n} \beta_j x_{ij} \tag{2.3}$$

where η_i is called the linear predictor in the jargon of generalized linear models, as it is a linear combination of the explanatory variables. It is connected to p_i by a non-linear (logarithm) so-called link function. The log-likelihood is

$$L = \sum_{i=1}^{m} y_i \log p_i + \sum_{i=1}^{m} (1-y_i) \log(1-p_i) \tag{2.4}$$

The penalized log-likelihood is

$$L^* = L - \frac{\lambda}{2} \sum_{j=1}^{n} \beta_j^2 \tag{2.5}$$

where λ is called the penalty parameter. The larger the λ, the stronger is its influence and the smaller the β_j^2's become. The value of λ can be determined by cross-validation. The ML method estimates the parameters by maximizing (2.5). Let $\mathbf{u}$ be a m-vector of ones, $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$, $\mathbf{p} = [p_1, p_2, \ldots, p_m]^T$, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots \beta_n]^T$ and $\mathbf{X}$ be a $m \times n$ matrix so that $\mathbf{X}(i, j) = x_{ij}$. Now we take the derivatives of L^* with respect to α and β_j so that:

$$\frac{\partial L^*}{\partial \alpha} = 0 \;\Rightarrow\; \mathbf{u}^T(\mathbf{y} - \mathbf{p}) = 0 \tag{2.6}$$

$$\frac{\partial L^*}{\partial \boldsymbol{\beta}} = 0 \;\Rightarrow\; \mathbf{X}^T(\mathbf{y} - \mathbf{p}) = \lambda\boldsymbol{\beta} \tag{2.7}$$

(2.6) and (2.7) are nonlinear because of the nonlinear relationship between $\mathbf{p}$, α and $\boldsymbol{\beta}$. To get a set of linear equations, we take the first order Taylor expansion of p_i,

$$p_i \approx \tilde{p}_i + \frac{\partial p_i}{\partial \alpha}(\alpha - \tilde{\alpha}) + \sum_{j=1}^{n} \frac{\partial p_i}{\partial \beta_j}(\beta_j - \tilde{\beta}_j) \tag{2.8}$$

where a tilde indicates an approximate solution. Now

$$\frac{\partial p_i}{\partial \alpha} = \tilde{p}_i(1 - \tilde{p}_i) \tag{2.9}$$

$$\frac{\partial p_i}{\partial \beta_j} = \tilde{p}_i(1 - \tilde{p}_i)x_{ij} \tag{2.10}$$

At the beginning of iterations, we have starting values about $\tilde{\alpha}$ and $\tilde{\boldsymbol{\beta}}$. Therefore, $\tilde{p}_i$ can be calculated from (2.3). Substituting (2.9) and (2.10) into (2.8), p_i can be represented by using linear combination of parameters α and $\boldsymbol{\beta}$. Substituting (2.8) into (2.6) and (2.7) and introducing $\tilde{w}_i = \tilde{p}_i(1 - \tilde{p}_i)$, $\tilde{\mathbf{W}} = \mathrm{diag}(\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_m)$, we have

$$\mathbf{u}^T\tilde{\mathbf{W}}\mathbf{u}\alpha + \mathbf{u}^T\tilde{\mathbf{W}}\mathbf{X}\boldsymbol{\beta} = \mathbf{u}^T(\mathbf{y} - \tilde{\mathbf{p}} + \tilde{\mathbf{W}}\tilde{\boldsymbol{\eta}}) \tag{2.11}$$

$$\mathbf{X}^T\tilde{\mathbf{W}}\mathbf{u}\alpha + (\mathbf{X}^T\tilde{\mathbf{W}}\mathbf{X} + \lambda\mathbf{I})\boldsymbol{\beta} = \mathbf{X}^T(\mathbf{y} - \tilde{\mathbf{p}} + \tilde{\mathbf{W}}\tilde{\boldsymbol{\eta}}) \tag{2.12}$$

where $\tilde{\boldsymbol{\eta}} = [\tilde{\eta}_1, \tilde{\eta}_2, \ldots \tilde{\eta}_m]^T$ and

$$\tilde{\eta}_i = \tilde{\alpha} + \sum_{j=1}^{n} \tilde{\beta}_j x_{ij} \tag{2.13}$$

for $i = 1, 2, \ldots, m$. Now (2.11) and (2.12) constitute a linear system and iterating with it generally leads to a solution quickly. In most cases, ten iterations are enough. Suitable starting values are $\tilde{\alpha} = \log[\bar{y}/(1-\bar{y})]$ with $\bar{y} = \sum_{i=1}^{m} y_i \Big/ m$ and $\tilde{\boldsymbol{\beta}} = 0$. If we introduce $\boldsymbol{\gamma}^T = [\alpha \,|\, \boldsymbol{\beta}^T]$ and $\mathbf{Z} = [\mathbf{u} \,|\, \mathbf{X}]$, (2.11) and (2.12) can be written as

$$(\mathbf{Z}^T \tilde{\mathbf{W}} \mathbf{Z} + \lambda \mathbf{Q}) \boldsymbol{\gamma} = \mathbf{Z}^T (\mathbf{y} - \tilde{\mathbf{p}} + \tilde{\mathbf{W}} \mathbf{Z} \tilde{\boldsymbol{\gamma}}) \tag{2.14}$$

where $\mathbf{Q}$ is a $(n+1) \times (n+1)$ identity matrix with $\mathbf{Q}(1,1) = 0$ to reflect that there is no penalty on α.

2.2.3 Partial Least Squares and Singular Value Decomposition

The linear system of (2.11) and (2.12) is huge: thousands of equations with an equal number of unknowns. Solving this could be computationally problematic and storing all the equations takes a substantial amount of memory space. PLS and SVD are both very popular dimension-reduction methods and they have been successfully applied to the field of gene expression based cancer classification. In this paper, both of these methods are proposed to undertake the task of solving (2.11) and (2.12). For an updated survey of PLS, interested readers can refer to this article (Wegelin, 2000). For the definition and computation of SVD, readers can refer to this book (Golub and van Loan, 1996). We would not go into the details of these two techniques here.

First, assume that the $m \times n$ matrix $\mathbf{X}$ stores all of the gene expression data with its rows being the microarray samples and its columns being the gene profiles. The formulations of PLS and SVD give the decomposition of $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{T}\mathbf{V}^T + \mathbf{R} \tag{2.15}$$

where $\mathbf{T}$ is a $m \times k$ matrix, $\mathbf{V}$ is a $n \times k$ matrix and $\mathbf{R}$ is a $m \times n$ matrix. k is the number of PLS components or singular values and $k \le m$. $\mathbf{R}$ is the residual matrix and can be considered as containing no useful information. Therefore, $\mathbf{X}$ can be approximated as

$$\mathbf{X} \approx \mathbf{T}\mathbf{V}^T \tag{2.16}$$

In PLS, the columns of $\mathbf{T}$ are called score vectors and the columns of $\mathbf{V}$ are called loading vectors. In SVD, $\mathbf{X}$ can be approximated using the first k singular values:

$$\mathbf{X} \approx \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{2.17}$$

where $\mathbf{U}$ is a $m \times k$ matrix, $\mathbf{S}$ is a $k \times k$ diagonal matrix and $\mathbf{V}$ is a $n \times k$ matrix. For convenience, let $\mathbf{T} = \mathbf{U}\mathbf{S}$ for SVD and use score vectors and loading vectors to name the columns of $\mathbf{T}$ and $\mathbf{V}$. Hence, we can use (2.16) to represent the decomposition of both PLS and SVD. The loading vectors produced by PLS and SVD are always mutually orthogonal and they are assumed to be normalized in PLS so that $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, which is a $k \times k$ identity matrix. Assume $\boldsymbol{\beta} = \mathbf{V}\boldsymbol{\theta}$ and substitute (2.16) into (2.11) and (2.12) and we have

$$\mathbf{u}^T\tilde{\mathbf{W}}\mathbf{u}\alpha + \mathbf{u}^T\tilde{\mathbf{W}}\mathbf{T}\mathbf{V}^T\mathbf{V}\boldsymbol{\theta} = \mathbf{u}^T(\mathbf{y} - \tilde{\mathbf{p}} + \tilde{\mathbf{W}}\tilde{\boldsymbol{\eta}}) \tag{2.18}$$

$$\mathbf{V}\mathbf{T}^T\tilde{\mathbf{W}}\mathbf{u}\alpha + (\mathbf{V}\mathbf{T}^T\tilde{\mathbf{W}}\mathbf{T}\mathbf{V}^T + \lambda\mathbf{I})\mathbf{V}\boldsymbol{\theta} = \mathbf{V}\mathbf{T}^T(\mathbf{y} - \tilde{\mathbf{p}} + \tilde{\mathbf{W}}\tilde{\boldsymbol{\eta}}) \tag{2.19}$$

Multiplying (2.19) by $\mathbf{V}^T$ we get

$$\mathbf{u}^T\tilde{\mathbf{W}}\mathbf{u}\alpha + \mathbf{u}^T\tilde{\mathbf{W}}\mathbf{T}\boldsymbol{\theta} = \mathbf{u}^T(\mathbf{y} - \tilde{\mathbf{p}} + \tilde{\mathbf{W}}\tilde{\boldsymbol{\eta}}) \tag{2.20}$$

$$\mathbf{T}^T\tilde{\mathbf{W}}\mathbf{u}\alpha + (\mathbf{T}^T\tilde{\mathbf{W}}\mathbf{T} + \lambda\mathbf{I})\boldsymbol{\theta} = \mathbf{T}^T(\mathbf{y} - \tilde{\mathbf{p}} + \tilde{\mathbf{W}}\tilde{\boldsymbol{\eta}}) \tag{2.21}$$

Remember $\tilde{\boldsymbol{\eta}} = \mathbf{Z}\tilde{\gamma}$ where $\mathbf{Z} = [\mathbf{u} \,|\, \mathbf{X}]$ and $\boldsymbol{\gamma}^T = [\alpha \,|\, \boldsymbol{\beta}^T]$, we have

$$\tilde{\boldsymbol{\eta}} = [\mathbf{u} \,|\, \mathbf{X}]\begin{bmatrix}\tilde{\alpha} \\ \tilde{\boldsymbol{\beta}}\end{bmatrix} = [\mathbf{u} \,|\, \mathbf{T}\mathbf{V}^T]\begin{bmatrix}\tilde{\alpha} \\ \mathbf{V}\tilde{\boldsymbol{\theta}}\end{bmatrix} = [\mathbf{u} \,|\, \mathbf{T}]\begin{bmatrix}\tilde{\alpha} \\ \tilde{\boldsymbol{\theta}}\end{bmatrix} \tag{2.22}$$

Redefine $\mathbf{Z} = [\mathbf{u} \,|\, \mathbf{T}]$ and $\boldsymbol{\gamma}^T = [\alpha \,|\, \boldsymbol{\theta}^T]$ so that we obtain

$$\tilde{\boldsymbol{\eta}} = \mathbf{Z}\tilde{\boldsymbol{\gamma}} \tag{2.23}$$

Thus the system of equations from (2.20) and (2.21) can also be represented by (2.14).

The length of $\boldsymbol{\theta}$ is k, the number of score vectors. Therefore, the total number of equations in (2.20) and (2.21) is $k+1$. Since $k \le m << n$, the order of the system of (2.11) and (2.12) is now effectively reduced from thousands to tens. Only a small amount of memory space is required and the equations can be solved quickly.

2.2.4 Feature Selection

Another very important part in cancer classification is to select a small subset of genes which can effectively separate different cancer types. RFE was proposed by (Guyon *et al.*, 2002) who combined SVM and RFE to select a small subset of genes to make accurate prediction for both acute leukemia and colon cancer data. Instead of evaluating the relevance of each gene independently, RFE tries to find a subset of genes which are most relevant with the cancers. To carry out this procedure, we need to know the ranks of the genes first:

$$\tilde{\boldsymbol{\beta}} = \mathbf{V}\tilde{\boldsymbol{\theta}} \tag{2.24}$$

where $\tilde{\boldsymbol{\beta}}$ gives the estimates of the regression coefficients and its absolute values indicate the relative importance of the genes in the subset. Now the RFE procedure is performed as follows:

1. For a subset of genes, leave-one-out cross-validation (LOOCV) is performed to find the λ which corresponds to the minimum LOOCV error.
2. 100 (or 50) bootstrap samples of the original data are generated. For each bootstrap sample, $\tilde{\boldsymbol{\beta}}$ is calculated with λ fixed. The averaged regression coefficients $\tilde{\boldsymbol{\beta}}$ are then calculated.

3. The genes with the smallest absolute values in $\tilde{\boldsymbol{\beta}}$ are eliminated. Therefore, a smaller subset of genes can be used for cancer classification.
4. The performance of the new subset of genes is evaluated by randomly partitioning the data into a training dataset and a testing dataset, for 100 (or 30) times. The averaged training and testing errors are recorded.

This procedure can be iterated for many times until there is only one gene left. An optimal subset of genes can be finally chosen.

The bootstrap is a non-parametric method for estimating the sampling distribution of a statistic. Given a sample dataset and a desired statistic (e.g., the mean), the bootstrap works by computing the desired statistic for a subsample of the dataset. The subsampling is done with replacement and the size of the sample is equal to the size of the original data set. The typical number of subsamples can be 50, 100, or even 1000. The more subsamples there are, the more stabilized estimate of the statistic can be obtained but the computational cost should also be considered. Based on experience, 50 or 100 subsamples can already give very stable estimates of the regression coefficients. So 100 bootstrap samples are chosen for breast cancer, central nervous system, colon tumor and acute leukemia datasets and 50 bootstrap samples for lung cancer, ovarian cancer and prostate cancer datasets, considering the sizes of the different datasets.

One difficulty of the performance evaluation of classifiers on microarray data is that the sample size is very small. To avoid any possible bias arising from a particular selection of testing samples, the original training and testing samples are combined and randomly partitioned into training and testing datasets of fixed sizes for a number of times. The means and the standard deviations of testing errors are then reported. Empirically, it is found that 30 or more random partitions can give a stable value of mean testing errors. Some statistical test like student T-test can be used to give the confidence level of the difference between the mean testing errors of two classifiers. Considering the computational costs of different dataset sizes, 100 random partitions are chosen for breast cancer, central nervous system, colon tumor and acute leukemia datasets and 30 random partitions are chosen for lung cancer, ovarian cancer and prostate cancer datasets.

2.2.5 Comparison with Least Squares Regression

Instead of giving the probability that the ith sample belongs to class 1, least-squares regression explicitly predicts the class label, y_i, of the ith sample. Usually in least-squares regression, one uses $y_i = 1$ or -1 to reflect that the ith sample belongs to class 1 or class -1. Least-squares regression aims to minimize the following cost function:

$$J = \sum_{i=1}^{m} \left(y_i - \alpha - \sum_{j=1}^{n} \beta_j x_{ij} \right)^2 \qquad (2.25)$$

where α, β_j are regression coefficients and x_{ij} denotes the jth gene expression level of the ith sample. The penalized least-squares regression (LSR) can be formulated as

$$J^* = J + \frac{\lambda}{2} \sum_{j=1}^{n} \beta_j^2 \qquad (2.26)$$

where λ is again the penalty parameter. Please note that LSR is used to denote penalized least-squares regression.

From a statistician's standpoint, logistic regression is very different from least-squares regression. The underlying mathematics is different and the computational details are different. Unlike the least-squares regression equations which can be solved by pseudoinversion, logistic regression equations are solved iteratively. A trial equation is fitted and then tweaked over and over until the improvement from one step to the next is suitably small.

From a practical standpoint, logistic regression and least-squares regression are almost identical. Both methods produce prediction equations. In both cases, the regression coefficients measure the predictive capability of the predictor variables. Least-squares regression is most suitable when the predictor variables are jointly multivariate normal and they have the same covariance for the two classes. Some researchers (Efron, 1975) pointed out that least-squares regression is more effective than logistic regression under this condition and requires less computational time. But in real applications, this condition is not usually satisfied. Press and Wilson (1978) have made an excellent discussion about choosing between logistic regression and least-squares regression and proved that in most practical applications, logistic regression excels least-squares regression.

There are other advantages about logistic regression. The prediction rule of logistic regression gives the probability of its prediction explicitly. This characteristic could be very important in medical diagnosis where an incorrect judgment can cause severe consequence. Another advantage of logistic regression is that the regression coefficients are easier to interpret. For example, the logistic regression coefficient for gene g is determined to be 0.06 and the prediction rule is

$$r = \exp(0.4 + 0.06g) \tag{2.27}$$

where $r = p/(1-p)$ is the odds of the sample being tumorous. Then if the variable of g is increased by one, the output is

$$\hat{r} = \exp(0.4 + 0.06(g+1)) \tag{2.28}$$

Dividing one equation by the other, we have

$$\frac{\hat{r}}{r} = \exp(0.06) = 1.062 \tag{2.29}$$

So the sample's odds of being tumorous increases by 6.2% if the expression level of gene g increases by one.

2.3 Results

2.3.1 Datasets

Seven publicly available cancer datasets were chosen from a public website (Li and Liu, 2002) to evaluate the performance of classifiers. The descriptions of the datasets are given as follows:

ALL-AML Leukemia (Golub *et al.*, 1999): This is a cDNA dataset. The original dataset contains 38 training samples and 34 testing samples over 7129 probes from 6817 human genes. These datasets contain measurements corresponding to ALL and AML samples from Bone Marrow and Peripheral Blood. Intensity values have been re-scaled such that overall intensities for each chip are equivalent. This is done by fitting a linear regression model using the intensities of all genes with "P" (present) calls in both the first sample (baseline) and each of the other samples. The inverse of the "slope" of the linear regression line becomes the (multiplicative) re-scaling factor for the current sample. This is done for every chip (sample) in the dataset except the baseline which gets a re-scaling factor of one.

Breast Cancer (Van't Veer *et al.*, 2002): This is an oligonucleotide dataset. The training data contains 78 patient samples, 34 of which are from patients who had developed distance metastases within 5 years (labeled as "relapse"), the rest 44 samples are from patients who remained healthy from the disease after their initial diagnosis for interval of at least 5 years (labeled as "non-relapse"). Correspondingly, there are 12 relapse and 7 non-relapse samples in the testing dataset. The number of genes is 24481 and the values of "Ratio" are extracted from original microarray data with the replacement of all "NaN" to 100.0.

Central Nervous System (Pomeroy *et al*., 2002): This is a cDNA dataset. Embryonal tumors of the central nervous system (CNS) represent a heterogeneous group of tumors about which little is known biologically, and whose diagnosis, based on morphologic appearance alone, is controversial. Only dataset C mentioned in the paper that is used to analyze the outcome of the treatment is provided on the website. Survivors are patients who are alive after treatment whiles the failures are those who succumbed to their disease. The data set contains 60 patient samples, 21 are survivors (labeled as "Class1") and 39 are failures (labeled as "Class0"). There are 7129 genes in the dataset.

Colon Tumor (Alon *et al*., 1999): This is an oligonucleotide dataset. It contains 62 samples collected from colon-cancer patients. Among them, 40 tumor biopsies are from tumors (labeled as "negative") and 22 normal (labeled as "positive") biopsies are from healthy parts of the colons of the same patients. Two thousand out of around 6500 genes were selected based on the confidence in the measured expression levels.

Ovarian Cancer (Petricoin *et al*., 2002): This is a proteomic mass spectrometry dataset. The goal of this experiment is to identify proteomic patterns in serum that distinguish ovarian cancer from non-cancer. This study is significant to women who have a high risk of ovarian cancer due to family or personal history of cancer. The proteomic spectra were generated by mass spectroscopy and the data set provided here is 6-19-02, which includes 91 controls (Normal) and 162 ovarian cancers. The raw spectral data of each sample contains the relative amplitude of the intensity at each molecular mass/charge (M/Z) identity. There are total 15154 M/Z identities. The intensity values were normalized according to the formula: NV = (V-Min)/(Max-Min), where NV is the normalized value, V the raw value, Min the minimum intensity and Max the maximum intensity. The normalization is done over all the 253 samples for all 15154 M/Z identities. After the normalization, each intensity value is to fall within the range of 0 to 1.

Prostate Cancer (Singh *et al*. 2002): This is an oligonucleotide dataset of tumor versus normal classification: training set contains 52 prostate tumor samples and 50 non-tumor (labeled as "Normal") prostate samples with around 12600 genes. An independent set of testing samples is from a different experiment and has a nearly 10-fold difference in overall microarray intensity from the training data. Besides, extra genes contained in the testing samples have been removed. In the above publication, the testing set is indicated to have 27 tumor and 8 normal samples. However, from the extraction, there are 25 tumor and 9 normal samples.

Lung Cancer (Gordon *et al*., 2002): This is an oligonucleotide dataset of classification between malignant pleural mesothelioma (MPM) and adenocarcinoma (ADCA) of the lung. There are 181 tissue samples (31 MPM and 150 ADCA). The training set contains 32 of them, 16 MPM and 16 ADCA. The rest 149 samples are used for testing. Each sample is described by 12533 genes.

All seven datasets were downloaded from the website mentioned above, which are already prepared in a uniform format. They can be directly used for classifier-training. Before applying the machine-learning methods, all gene expressions were log-transformed and normalized to have zero means and unit variances. In the following, the original partitioning of training and testing data is not followed and they are simply combined and randomly repartitioned for the purpose of evaluation.

2.3.2 Classification Accuracy

A linear SVM was also used to compare with the two regression methods. Two software packages were used to implement the SVM. The first MATLAB package was written by Schwaighofer (2001). Though it can run fast, it seems to have stopping problem when running on the prostate cancer dataset. So another software package written by Gunn (2001) was used for testing the prostate cancer data. The regularization parameter, C, of SVM is chosen by cross-validation so that the cross-validation error is minimum. The value of $\log_2 C$ is set to increase from -6 to 13 with a step-size of one. The specifications of the seven cancer datasets are listed in Table 2.1. The sizes of training and testing datasets were chosen arbitrarily. However, the sizes of training datasets were set to be large enough for classifier training, and the size of training dataset was set to be always larger than the size of testing dataset. For each of these datasets, the evaluation of the classifier is given as follows: 100 random partitions are performed and the dataset is separated into a training dataset and a testing dataset with pre-specified sizes. The testing errors of all partitions were recorded and their means and standard deviations were calculated. The smaller mean testing errors that a classifier achieves, the better classifier it is.

At first, all of the SVD components are used for both PLR and LSR. So both the regression methods were evaluated without data loss. Because the number of SVD

Table 2.1 Description of the seven cancer datasets.

Dataset	Genes	Partition Setting
Breast Cancer	24481	60 trainings vs. 37 testings
Central Nervous System	7129	40 trainings vs. 20 testings
Colon Tumor	2000	40 trainings vs. 22 testings
Acute Leukemia	7129	40 trainings vs. 32 testings
Lung Cancer	12533	100 trainings vs. 81 testings
Ovarian Cancer	15154	150 trainings vs. 103 testings
Prostate Cancer	12600	100 trainings vs. 36 testings

components equals the rank of the gene expression data matrix, all noises and redundancy of gene expression data are included in the estimation of the regression coefficients. The results are listed in Table 2.2. It can be seen that PLR performs better than LSR.

Now the results using only 15 PLS and SVD components are listed in Table 2.3. The number 15 is determined empirically and it was considered sufficient for all the datasets that were used. PLR and LSR based on 15 SVD components performed equivalently with PLR and LSR based on all SVD components. However, by 15 PLS components, the results of the regression methods are enhanced in both accuracy and efficiency. Comparing with the results of SVM, the regression methods have shown very competitive performance. The PLR also shows superiority over the LSR whether PLS or SVD is used for dimension-reduction. Comparing PLS and SVD, the PLS based classifier generally excels except on the lung cancer dataset. It is illustrated in Section 2.3.6 that for PLS, less components are required for classification than those of SVD. Because PLS incorporates class labels during the dimension-reduction process, usually it produces more informative components than SVD for classification purpose.

Table 2.2 Classification results using all SVD components. Testing error means (μ) and standard deviations (σ) are listed.

Dataset	PLR μ,σ	LSR μ,σ
Breast Cancer	13.55, 2.72	14.3, 2.82
Central Nervous System	8.17, 1.67	8.19, 1.79
Colon Tumor	4.35, 1.80	4.62, 1.60
Acute Leukemia	2.84, 1.83	2.88, 1.83
Lung Cancer	0.76, 0.70	0.92, 0.76
Ovarian Cancer	1.26, 0.67	1.35, 0.58
Prostate Cancer	5.45, 1.87	5.67, 1.89

Table 2.3 Classification results using 15 PLS and SVD components. Testing error means (μ) and standard deviations (σ) are listed.

Dataset	PLR		LSR		SVM
	PLS μ,σ	SVD μ,σ	PLS μ,σ	SVD μ,σ	μ,σ
Breast Cancer	12.88, 2.89	13.50, 2.58	13.80, 2.82	14.43, 2.87	13.09, 2.70
Central Nervous System	7.68, 1.76	8.14, 2.58	8.04, 2.03	8.20, 1.69	7.76, 2.06
Colon Tumor	3.97, 1.62	4.26, 1.55	4.26, 1.75	4.62, 1.42	3.78, 1.23
Acute Leukemia	1.94, 1.82	2.74, 1.88	2.48, 1.83	2.85, 1.95	1.57, 1.33
Lung Cancer	0.83, 0.71	0.52, 0.61	1.12, 0.76	0.73, 0.61	0.83, 0.82
Ovarian Cancer	0.08, 0.42	1.29, 1.26	0.33, 0.56	1.31, 0.96	0.22, 0.50
Prostate Cancer	4.68, 1.77	5.38, 1.75	5.49, 1.88	10.49, 2.41	4.75, 1.51

Table 2.4 Computational time of PLR, LSR and SVM.

Dataset	PLR (s)		LSR (s)		SVM(s)
	SVD	PLS	SVD	PLS	
Breast Cancer	5656	4602	4602	4455	32615
Central Nervous System	924	930	834	814	2297
Colon Tumor	419	404	300	289	1577
Acute Leukemia	1210	937	827	822	2363
Lung Cancer	7565	5443	6056	5409	23245
Ovarian Cancer	17010	14962	16850	14647	37931
Prostate Cancer	7200	5419	6205	5136	180070

2.3.3 Computational Time

The computational time of the above experiments are listed in Table 2.4. Generally, PLS based classifier uses less time than SVD based classifier. The PLR training requires solving a set of linear equations iteratively until convergence while the LSR training requires solving a set of linear equations only once. So it is reasonable to see that PLR uses more time than LSR. Nevertheless, it can also be seen that PLR does not cost "significantly" more time than LSR. That means, the dimension-reduction method itself takes the main part of the total time cost, and the time cost of classifier training and testing is greatly reduced by using a dimension-reduction method. SVM costs most time among all the methods used. The training time of SVM is affected by both the number of samples and the number of variables.

The time to seek support vectors for SVM also depends on the characteristics of different datasets. The large number of variables does not add too much to the time for dimension-reduction, and the training time for the regression methods depends on the number of samples and the very small number of components. Therefore, the overall time required by PLS and SVD based regression methods can be much less than that of SVM in this "small m, large n" condition. It can also be seen that the SVM algorithm written by Gunn (2001) costs much more time than that written by Schwaighofer (2001) because the computational time on the prostate cancer dataset is significantly larger than the computational time on all the other datasets.

2.3.4 Choosing Penalty Parameter

The acute leukemia dataset is used. It consists of two classes: acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). Each sample has 7129 genes. The original dataset (Golub et al., 1999) was separated into a 38-training set and a 34-testing set. So there are totally 72 samples (47 ALL and 25 AML) in the dataset. Here, PLS and SVD based PLR are designated as PLS-PLR and SVD-PLR respectively for convenience.

The estimates of the regression coefficients $\tilde{\boldsymbol{\beta}}$ are affected by the penalty parameter, λ, significantly. So choosing a proper value for λ is crucial for classifier training. A direct way is to select the penalty parameter by cross-validation. To find an optimal value of λ, it was varied in steps over a large range: $2^{-15}-2^{15}$, using 31 linearly spaced values for $\log_2 \lambda$. Figure 2.1 shows the LOOCV errors of acute leukemia data against $\log_2 \lambda$. For both PLS-PLR and SVD-PLR, the minimum LOOCV error is 3 out of 72, which happens when $\log_2 \lambda$ is relatively small: from -14 to -7. The small value chosen for λ can be attributed to the use of dimension-reduction methods, which already remove some degree of redundancy. The LOOCV error becomes maximum when $\log_2 \lambda \geq 0$. The maximum LOOCV error turns out to be 25, which is the number of total AML samples in the acute leukemia dataset. That means, the PLR classifier totally lost its prediction ability if λ is improperly set to a large value. The curves of PLS-PLR and SVD-PLR appear to be very similar in Figure 2.1. The values of log-likelihood are also shown in Figure 2.2. The minimum log-likelihood is nearly zero for both PLS-PLR and SVD-PLR, which corresponds to the minimum LOOCV error. The minimum log-likelihood that can be achieved in a logistic regression is zero. Therefore, it indicates a successful training has been done.

One advantage of logistic regression is that it produces the probabilities of its prediction directly. Figure 2.3 shows the probabilities of prediction for both ALL and AML samples. The results of PLS-PLR and SVD-PLR are given in Figure 2.3(a) and (b), respectively. The probability, p, indicates the chance that a sample belongs to ALL based on its gene expression levels. Hence, the probability that a sample belongs to AML is $1-p$. When λ becomes large, the probabilities of all samples converge to a fixed value, which equals the percentage of ALL samples in all acute leukemia samples. This does not happen by chance; that is because $\tilde{\alpha} = \log[\bar{y}/(1-\bar{y})]$ and $\bar{y} = \sum_{i=1}^{m} y_i \big/ m$ are set as the initial values. Another thing which should be noticed is that the curves for all ALL and all AML samples in Figure 2.3(a) overlap themselves, respectively. It means that the probabilities given by PLS-PLR have a much smaller deviance than those of SVD-PLR.

2.3.5 Recursive Feature Elimination

Feature selection has been done on the seven cancer datasets using RFE. Assume that the number of genes in one gene subset is n. The optimal way to do RFE

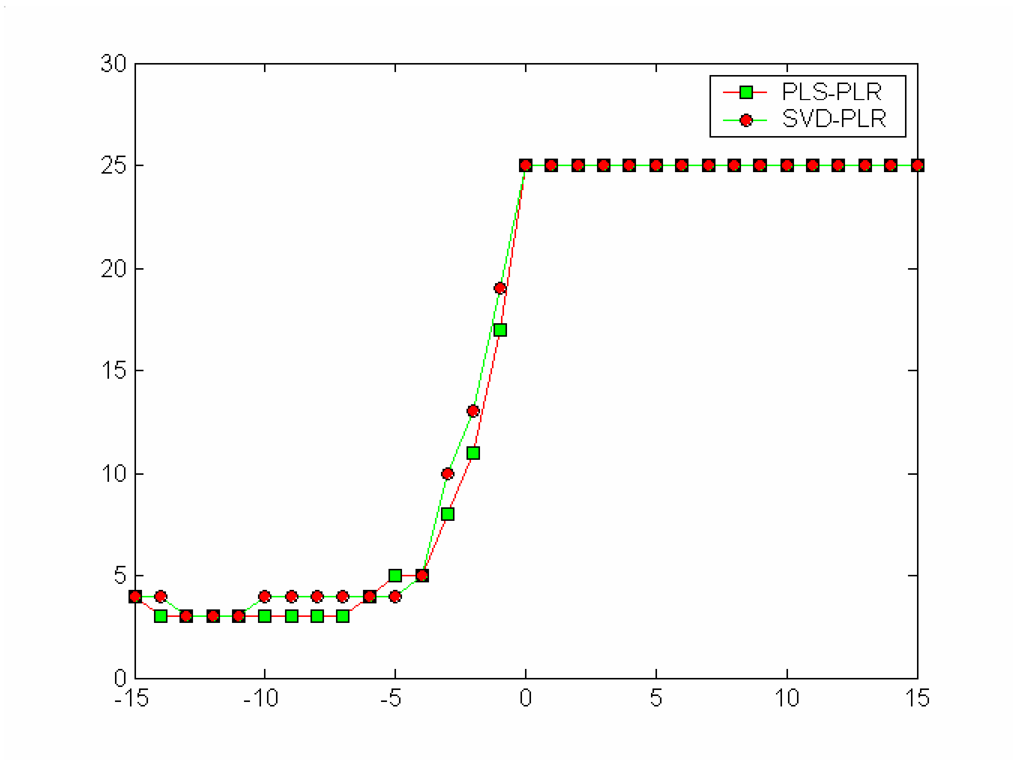


Figure 2.1 LOOCV errors vs. $\log_2 \lambda$ on 72 acute leukemia samples for both PLS-PLR and SVD-PLR.

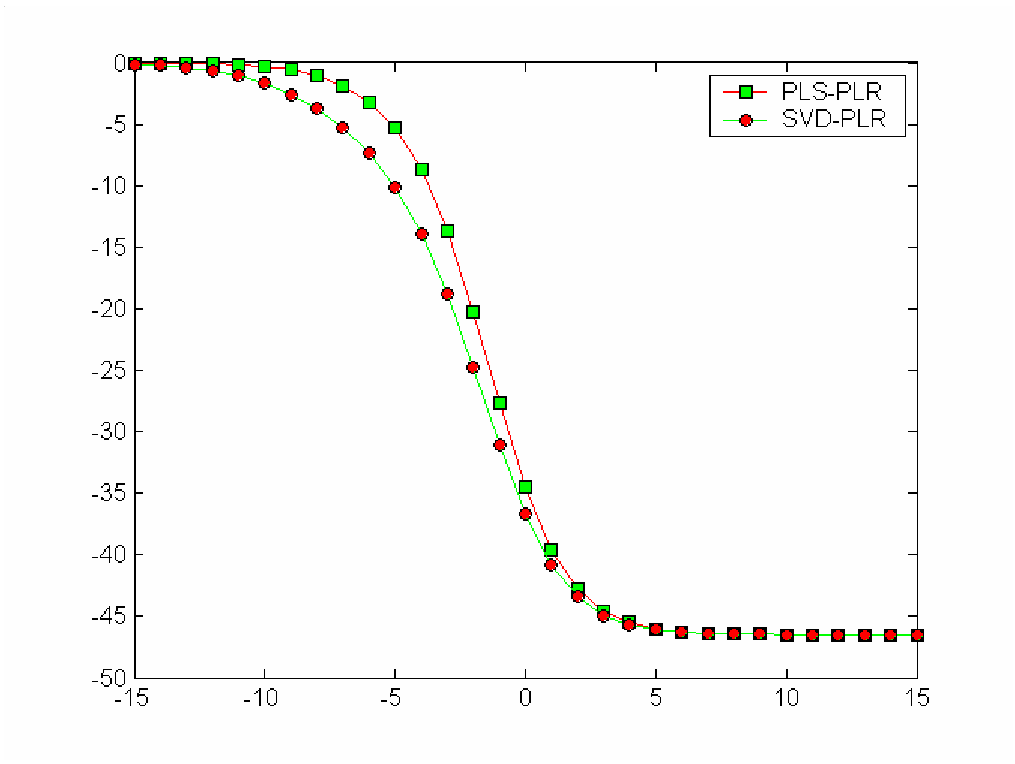
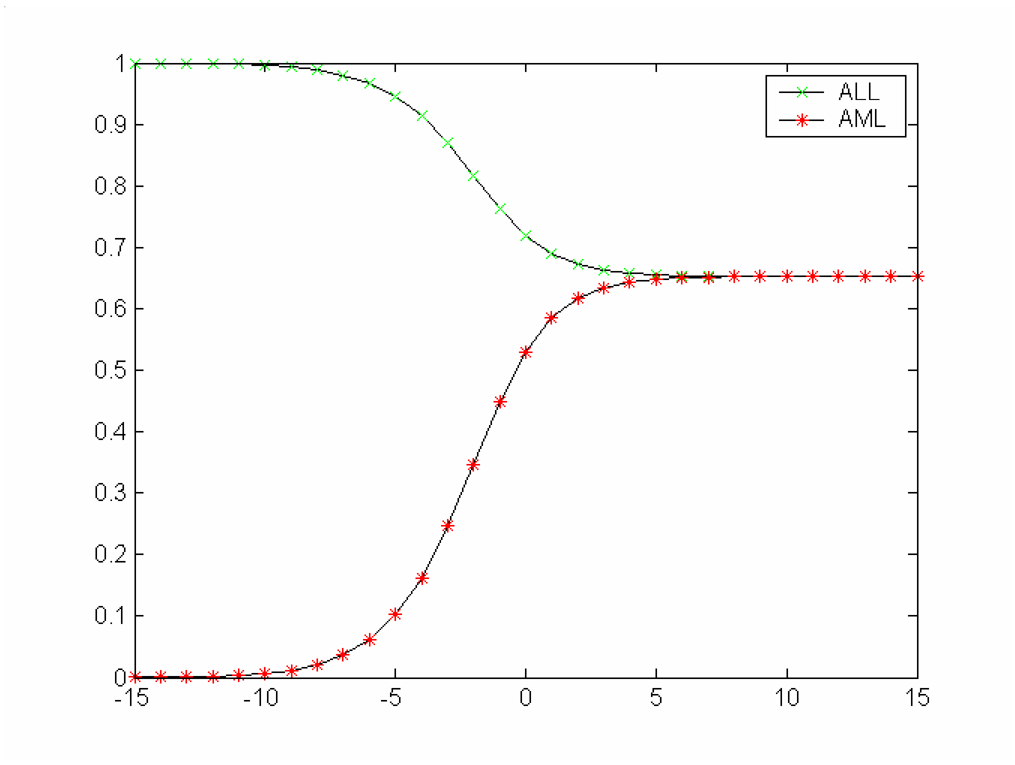
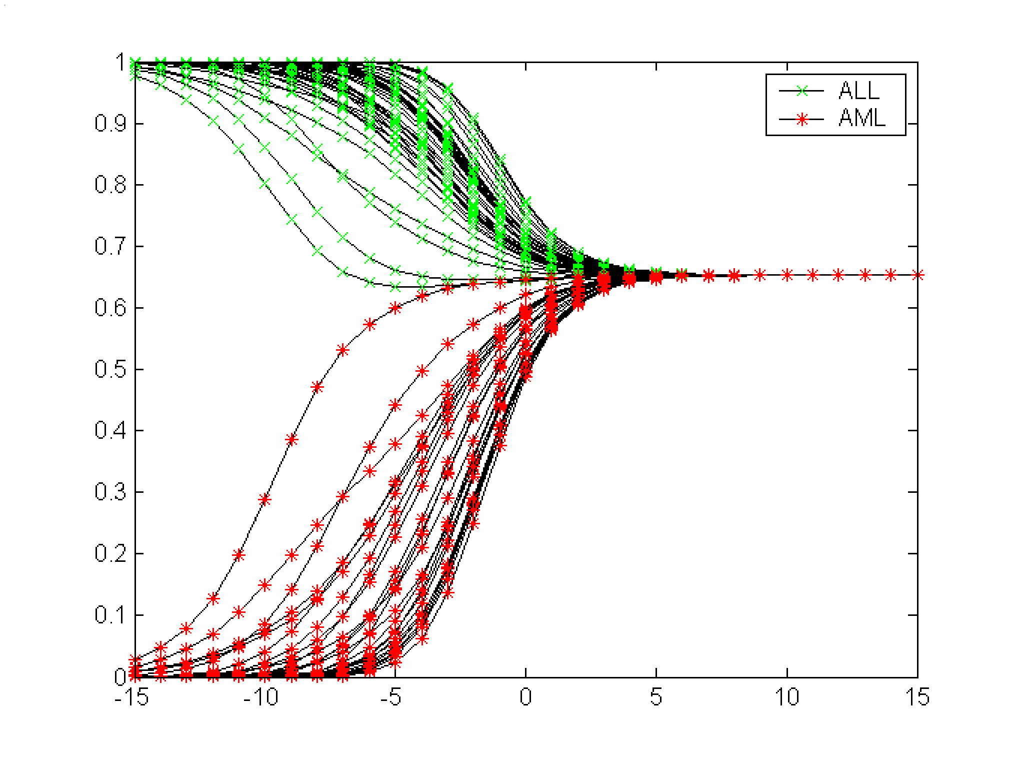


Figure 2.2 Log-likelihood vs. $\log_2 \lambda$ on acute leukemia cancer data for both PLS-PLR and SVD-PLR.



(a)

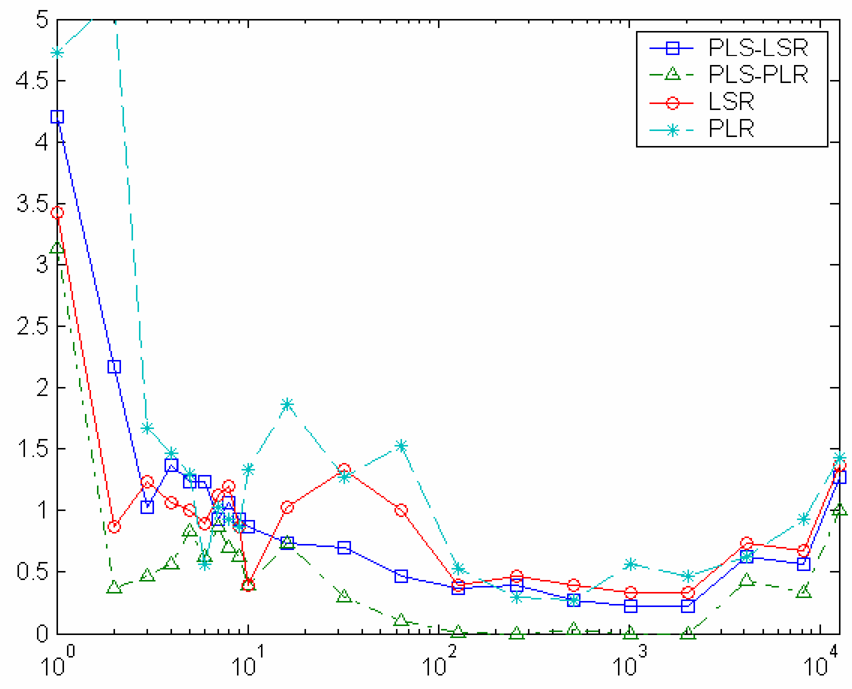


(b)

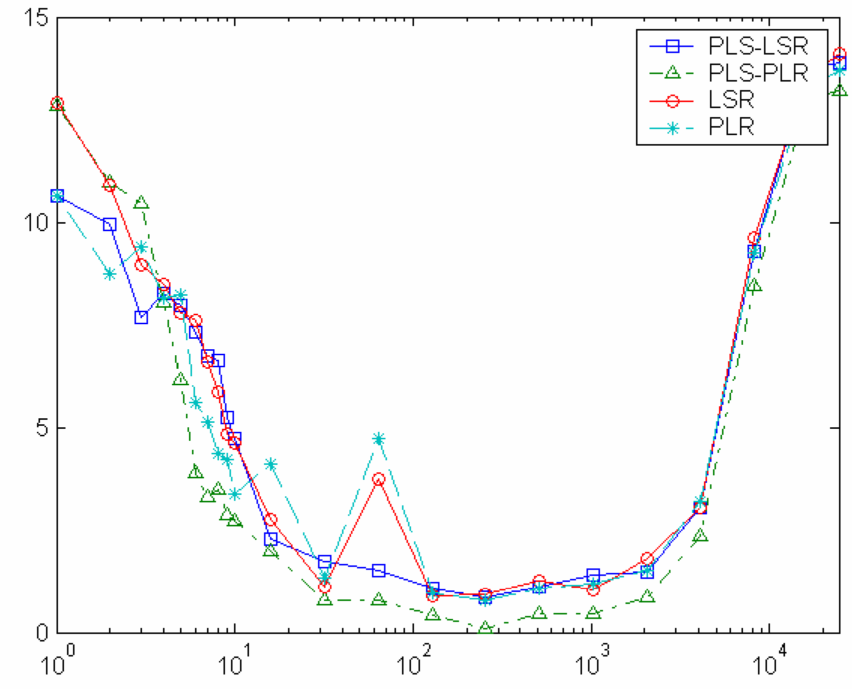
Figure 2.3 Probability outputs vs. $\log_2 \lambda$ on acute leukemia cancer data. (a) PLS-PLR. (b) SVD-PLR.

is to eliminate the least important gene one at each time of feature elimination but this will need enormous time to complete. An eclectic method is to eliminate a large amount of genes at each time when n is large and less amount of genes when n becomes small. The RFE procedure is designed in the following way. n is set to be fixed for each subset; thus a series of nested subsets can be obtained. The value of n would be halved each time when the subset is reduced until it is less than 10; then one gene is eliminated at each time. For example, the colon cancer data contains 2000 gene expression levels. Then the subsets of RFE for colon data would be: 2000, 1024, 512, 256, 128, 64, 32, 16, 10, 9, 8, …, 1. RFE was performed with four classifiers: PLS based PLR (PLS-PLR), PLS based LSR (PLS-LSR), PLR using all SVD components (PLR), and LSR using all SVD components (LSR). Therefore, RFE combined with classifiers with or without data reduction are both considered. The mean testing errors of the four classifiers are plotted in Figure 2.4 (a)-(g). The two curves from the same dimension-reduction method look more similar than the two curves from the same regression method. Therefore, it seems that the dimension-reduction methods rather than the regression methods have more influence on the process of feature selection. Generally, the curve of PLS-PLR is below the other three curves, meaning that its testing errors are smaller.

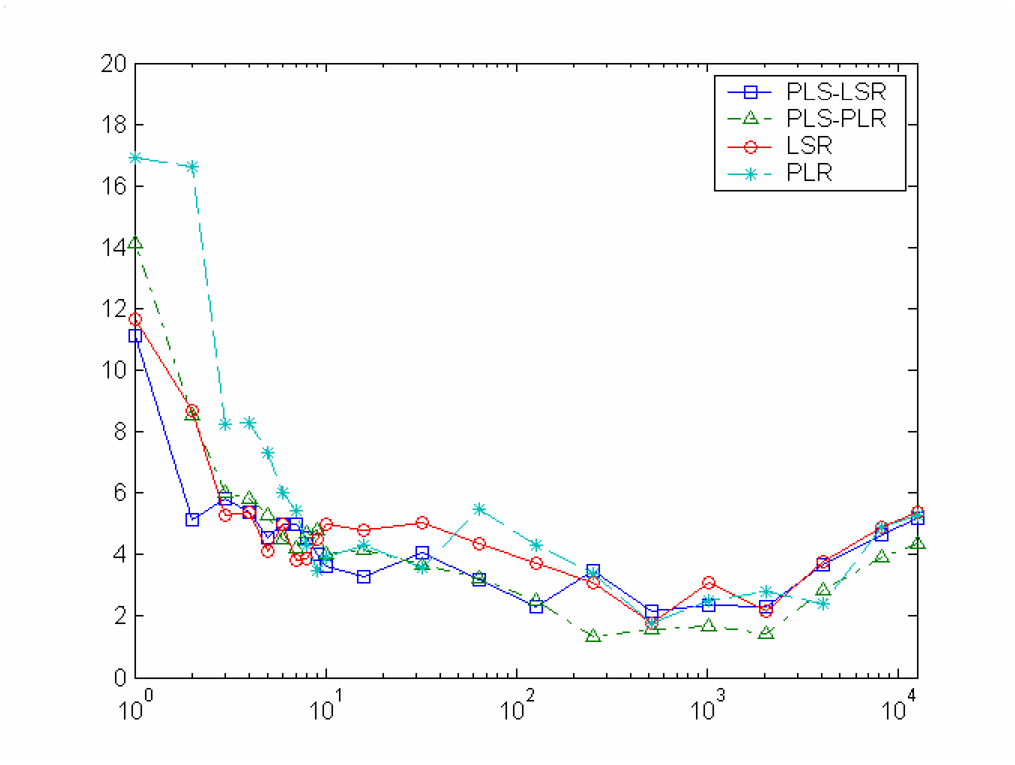
The testing errors of breast, prostate, colon and central nervous systems cancer data have been significantly reduced when less features are used than the original feature set. Almost perfect classification can be achieved on all the seven cancer datasets by combining RFE with PLR methods. However, we can also observe that for lung, acute leukemia and ovarian cancer data, testing errors are not very sensitive to the number of features.


PLS-LSR
PLS-PLR
LSR
PLR
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
10^0
10^1
10^2
10^3
10^4

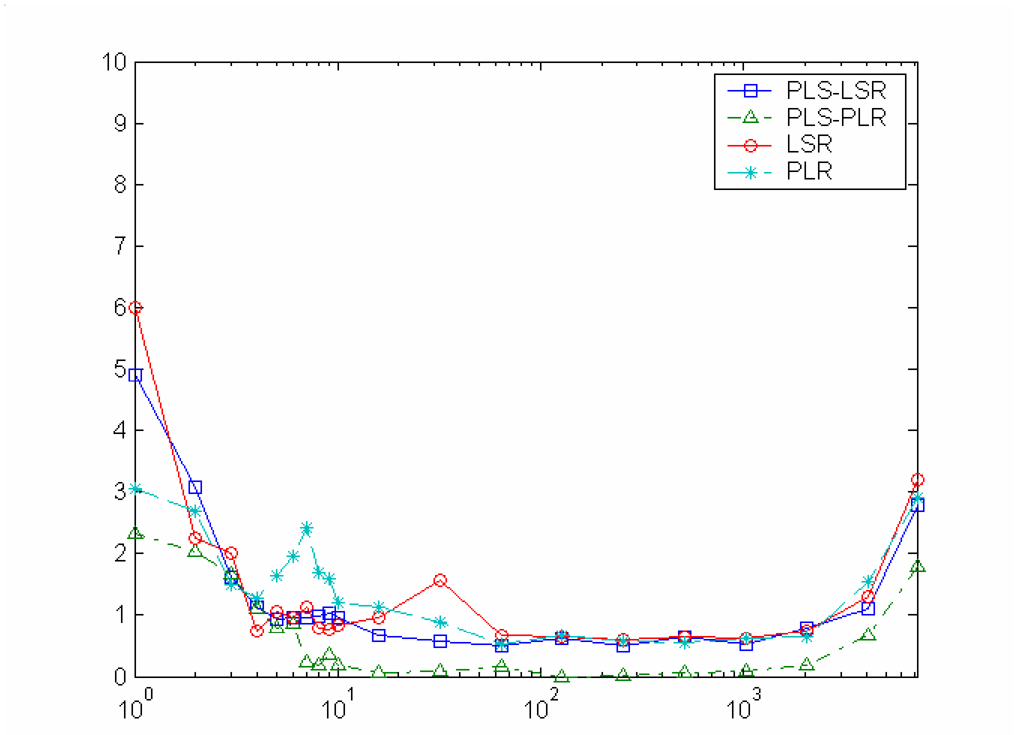
(a)


PLS-LSR
PLS-PLR
LSR
PLR
0
5
10
15
10^0
10^1
10^2
10^3
10^4

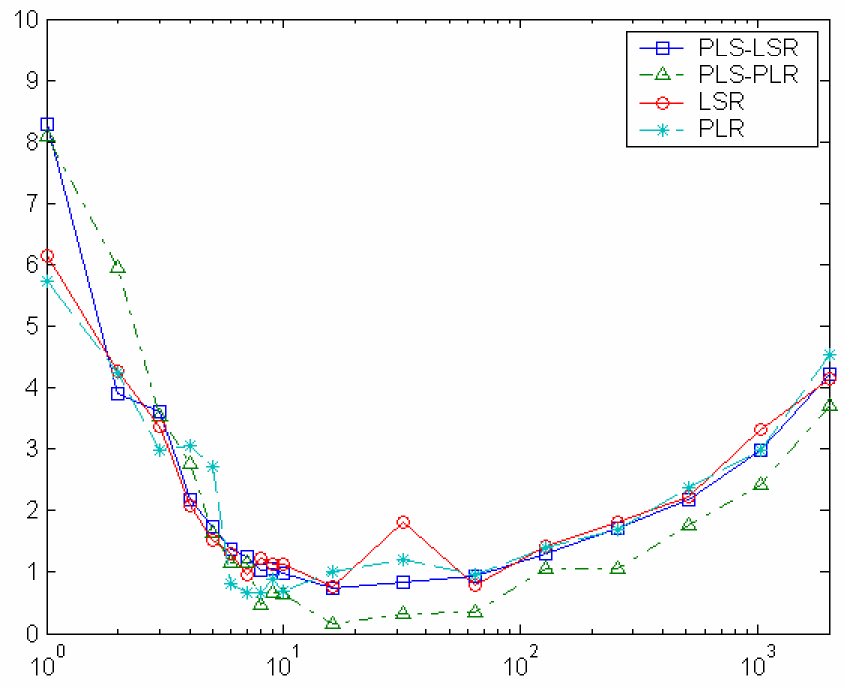
(b)


PLS-LSR
PLS-PLR
LSR
PLR
20
18
16
14
12
10
8
6
4
2
0
10^0
10^1
10^2
10^3
10^4

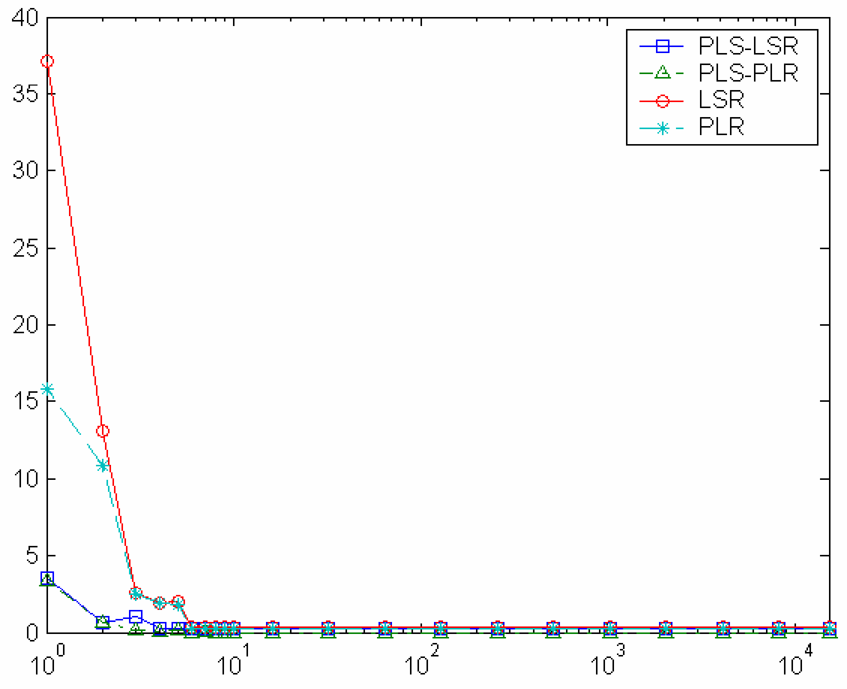
(c)


PLS-LSR
PLS-PLR
LSR
PLR
10
9
8
7
6
5
4
3
2
1
0
10^0
10^1
10^2
10^3

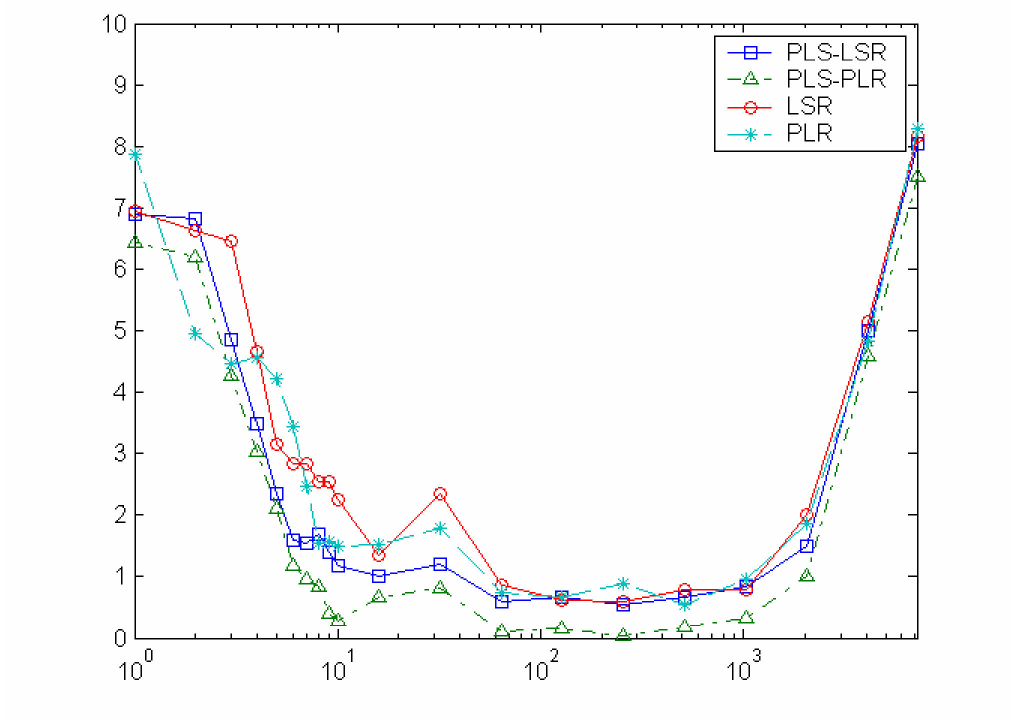
(d)



(e)



(f)



(g)

Figure 2.4 Averaged testing errors vs. sizes of gene subsets. RFE using four classifiers: PLS-PLR, PLS-LSR, PLR using all SVD components (PLR) and LSR using all SVD components (LSR). (a) Lung cancer. (b) Breast cancer. (c) Prostate cancer. (d) Acute leukemia. (e) Colon cancer. (f) Ovarian cancer. (g) Central nervous systems.

Table 2.5 List of subsets with minimum mean testing errors.

Dataset	Description of subsets	
	Features	Accuracy (%)
Breast Cancer	256	99.8
Central Nervous System	256	99.8
Colon Tumor	16	99.3
Acute Leukemia	128	100
Lung Cancer	256	100
Ovarian Cancer	6	100
Prostate Cancer	256	96.3

The subsets with the minimum mean testing errors are summarized in Table 2.5. It shows that highly accurate classification can be made on much smaller feature subsets than the original datasets. Almost perfect classification accuracy is achieved on six of the above cancer datasets. Only for prostate cancer, the highest accuracy reported is 96.3%. With reference to Singh *et al.* (2002) for the result of prostate cancer, it reported a 90% LOOCV accuracy on the training dataset using 4 or more genes and 86% testing accuracy using a 16-gene model. The k -nearest neighbor algorithm was used as the classification method. Tan and Gilbert (2003) reported a 73.53% testing accuracy using bagging C4.5 decision tree method on the same dataset. Dettling and Bühlmann (2004) also used PLR to achieve about 91% averaged testing accuracy on 50 random partitions of the same dataset. However, they used gene subset in a forward feature selection fashion. Therefore, our results on the prostate cancer dataset are very good and quite competitive.

2.3.6 Components Selection

Another important issue for PLS-PLR and SVD-PLR is choosing the number of components used for training and testing. Because there are always noises in microarray cancer data, the maximum number of components that can be extracted from PLS and SVD is usually equal to the number of samples in the dataset. It is not necessary to use all of these components. Since these components are always sorted according to their variances in descending order, only the first few components are needed and the other components can be considered as noises. To determine the effect of components selection, we set $\lambda = 0$ and then perform LOOCV on the acute leukemia data while the number of components varies from 1 to 20. Figure 2.5 shows the findings. PLS-PLR achieves the minimum LOOCV error using only five components. It can even reach four LOOCV errors using only two components. More components used than five did not help PLS-PLR to make better results. For SVD-PLR, the minimum LOOCV error appears when 10 components are used. The results

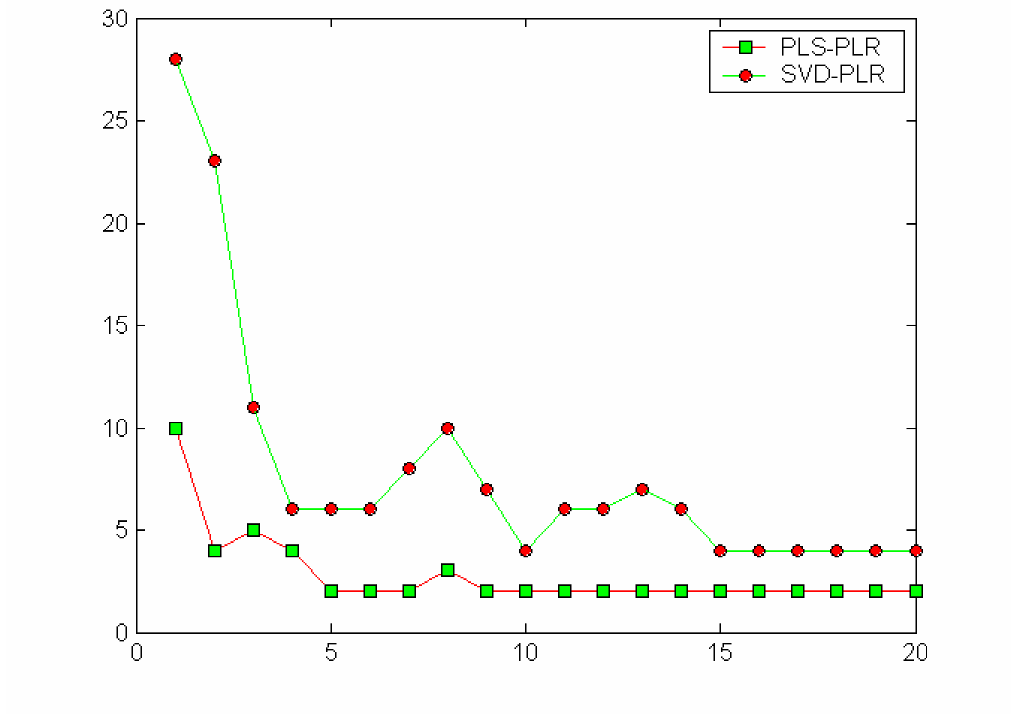
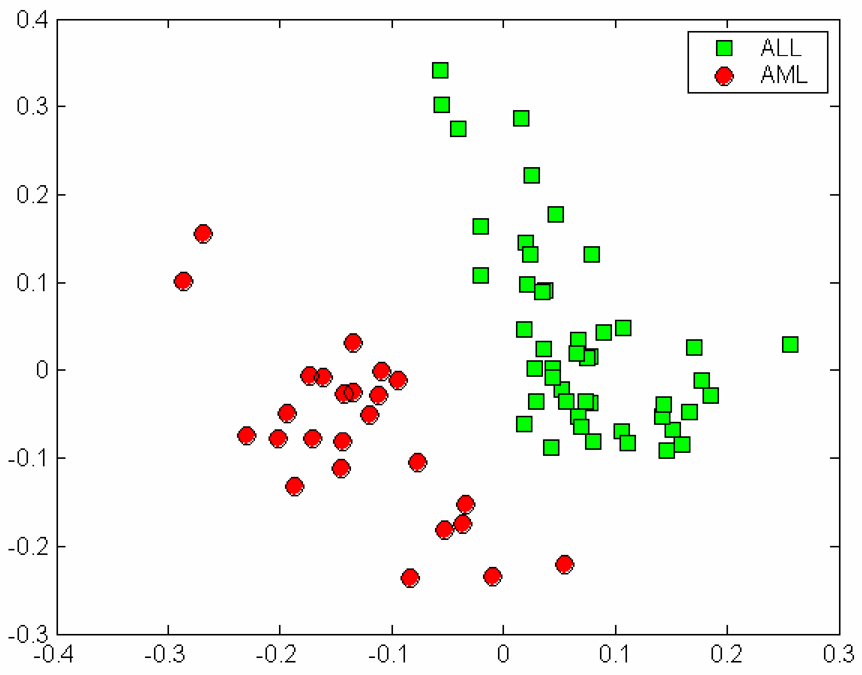
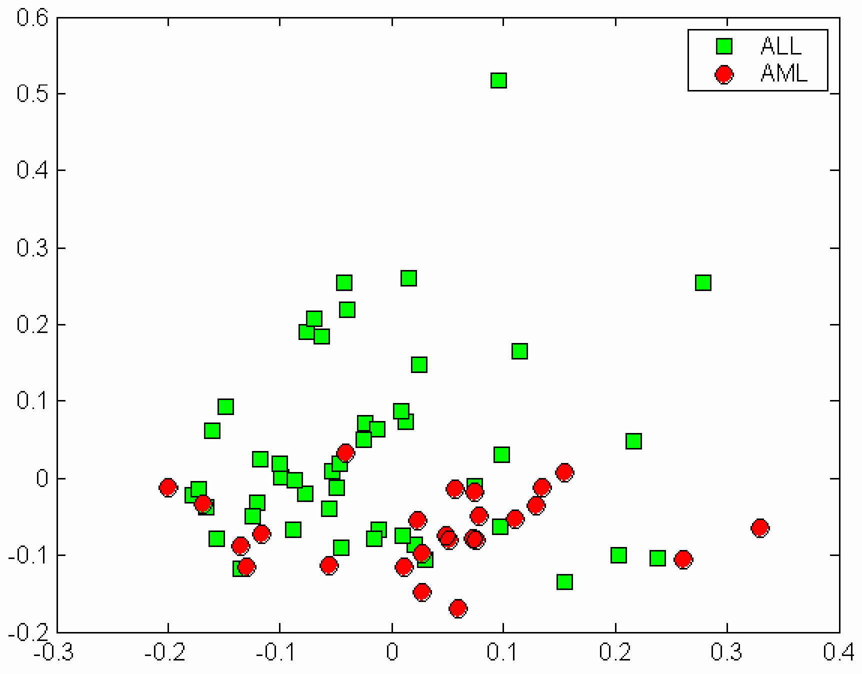


Figure 2.5 LOOCV errors vs. number of components used. PLS-PLR and SVD-PLR are applied on 72 acute leukemia samples.

of LOOCV begin to stabilize when 15 and more components are used for SVD-PLR. It is convincing that PLS produces components that are of "better quality" than those of SVD from this comparison. The advantage of PLS components against SVD components can be further shown in Figure 2.6(a) and (b), where all samples are plotted using the first two components from PLS and SVD, respectively. Two PLS components are enough to nicely separate all 72 acute leukemia samples while the two clusters in the plot of SVD components overlap heavily.



(a)



(b)

Figure 2.6 Scatter plot of 72 acute leukemia samples using two components from: (a) PLS. (b) SVD.

2.4 Discussions and Summary

Penalized logistic regression combined with PLS has shown excellent performance on microarray data for cancer classification. Comparing with the state-of-the-arts method, SVM, its classification accuracy is quite competitive. Because PLS can be executed very efficiently on the microarray data, the overall computational time for PLS-PLR is much less than the time by SVM. It is also found that the PLR is more superior to the LSR for cancer classification. The PLR is more robust and can perform very well no matter whether the original data are jointly multivariate or have the same covariance matrix.

Explicit output of prediction strength (probability) is another advantage of logistic regression. It can help pathologists to make better final judgments or to take further investigations. Logistic regression also makes the interpretation of regression coefficients much easier.

PLS was first advocated by researchers in chemometrics where the data also have the condition of "small m , large n ". Later some researchers (Nguyen and Rocke, 2002) have proved its superiority as a dimension-reduction method for microarray data based cancer classification against SVD. In this work, its advantage is shown as a solution for PLR. Therefore, the conclusion is coherent with the findings of other researchers. The number of components used by PLS is also more stable than that used by SVD. Sometimes the SVD components may not contain useful information for separating different tumor types, and more components have to be included into training so that the classifier is able to recognize the patterns of gene expression levels of various cancers.

Some authors view dimension-reduction method as an alternative to feature selection and call it "feature re-construction". However, it is found that the PLS and

SVD based PLR can also benefit from feature selection. That is because the redundant features may prohibit dimension-reduction methods from correctly extracting components for classification. Therefore, the dimension-reduction methods here are used to further reduce the dimension and remove noises and redundancy from the data of selected features. With the orthogonality property and the small number of components generated from dimension-reduction methods, the time required for PLR training is also reduced.

Chapter 3. Kernel Partial Least Squares and Kernel Principal Component Analysis Based Regularized Least-Squares Classification for Cancer Classification

3.1 Introduction

In the context of microarray, even for regularized kernel classifiers, like support vector machine (SVM) or regularized least-squares classification (RLSC), which can handle high-dimensional data, the existence of irrelevant genes and noise may still harm the performance of the classifiers.

This naturally calls for the application of dimension reduction methods. Some previous work is cited in the following in chronical order: West *et al*. (2001) have combined singular value decomposition (SVD) with Bayesian regression; Ghosh (2002) has combined SVD with logistic regression; Nguyen and Rocke (2002) have combined partial least squares (PLS) with logistic discrimination (LD) and quadratic discriminant analysis (QDA); Stephanopoulos *et al*. (2002) have combined Fisher discriminant analysis (FDA) with nearest group centroids; Antoniadis *et al*. (2003) have combined effective dimension reduction methods with LD, linear discriminant analysis (LDA) and QDA; Ghosh (2003) has combined PLS and PCA with LDA; Huang and Pan (2003) have used PLS and penalized PLS (PPLS) directly for classification; Lilien *et al*. (2003) have combined PCA with LDA; Fort and Lacroix (2005) have combined different variants of PLS with penalized logistic regression (PLR); Shen and Tan (2005a) have combined PLS and PCA with PLR. All these methods have achieved very good performance. However, they have considered only the traditional dimension-reduction methods with conventional statistical classifiers. Whether or not the more modernized kernel dimension-reduction methods, like kernel partial least squares (KPLS) (Rosipal and Trejo, 2001) and kernel principal component analysis (KPCA) (Scholköpf, *et al*., 1998), can be combined with the state-of-the-art regularized kernel classifers to enhance cancer classification performance has not been systematically answered.

Among the kernel machine learning classifiers, SVM (Vapnik, 1998) has gained popularity because of its superior performance compared with other classifiers across many domains and problems. A recent study has also shown its superiority for both binary and multiclass cancer classification (Statnikov *et al*., 2005). SVM is one of the regularized kernel classifiers which can be derived from Tikhonov regularization approach (Rifkin and Klautau, 2004). However, another classifier called RLSC from the same category of kernel classifiers has also been proven to perform equally well with SVM (Rifkin, 2002). Therefore, the choice between SVM and RLSC should be made on computational tractability instead of the accuracies of the two classfiers. RLSC is also known in literature as least squares-SVM (LS-SVM) (Suykens and Vandewalle, 1999) or Proximal SVM (Fung and Mangasarian, 2001).

In this chapter, two kernel dimension-reduction methods, KPLS and KPCA, are considered. After dimension reduction, a linear RLSC is followed up for classification. Thus two classifiers, KPLS+RLSC and KPCA+RLSC have been derived and tested on several cancer datasets. It has been illustrated that by using mutual orthogonality of the components from kernel dimension reduction, a very simple solution can be derived for the linear RLSC, which only requires the inversion of a sparse and low-dimensional matrix. The two algorithms have also been shown to have excellent performance in comparison with SVM. Several other issues regarding the usage of KPLS and KPCA are discussed and should be valuable when combining them with other classification methods.

3.2 Methods

3.2.1 Regularized Kernel Classifiers

Given a microarray dataset containing m samples, with each sample represented by the expression levels of n genes, they can be denoted as $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)$, where $\mathbf{x}_i$ is a n-length vector with its entries being gene expression levels; y_i is the class label of the ith sample with $y_i = 1$ or -1 indicating two different classes. The goal of binary cancer classification is to design a classifier to separate the samples into two different classes (tumorous vs. normal; type A vs. type B; subtype C vs. subtype D) given the gene expression (or proteomic pattern) data.

The Tikhonov regularization method is introduced first, from which a broad class of classifiers can be derived. In reproducing kernel Hilbert space (RKHS), the Tikhonov regularization is the minimization problem (Rifkin, 2002):

$$\min_{f \in \mathrm{H}} \sum_{i=1}^{m} \mathrm{V}(f(\mathbf{x}_i), y_i) + \lambda \|f\|_{\mathrm{K}}^2 \tag{3.1}$$

where $\mathrm{V}(f(\mathbf{x}), y)$ is the loss incurred when the actual label of sample $\mathbf{x}$ is y; λ is the regularization parameter. If $\mathrm{V}(f(\mathbf{x}), \mathbf{y}) = (f(\mathbf{x}) - \mathbf{y})^2$ is chosen, the so called "squared loss", the Tikhonov minimization problem becomes the RLSC. If $\mathrm{V}(f(\mathbf{x}), y) = (1 - yf(\mathbf{x}))_+ = \max(0, 1 - yf(\mathbf{x}))$ is chosen, the so called "hinge loss", the Tikhonov minimization problem becomes the well-known SVM. According to the representer theorem (Schölkopf and Smola, 2002), the solution to the Tikhonov minimization problem can be written as a sum of kernel products on the training dataset:

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i \mathrm{K}(\mathbf{x}, \mathbf{x}_i) + b \tag{3.2}$$

under very general conditions on the loss function V, where b is the unregularized bias term. So the goal is to find the values for $c_1, c_2, \ldots, c_m$ and b. Let $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$ and $\mathbf{c} = [c_1, c_2, \ldots, c_m]^T$; it can be derived that for RLSC (Appendix A), $\mathbf{c}$ and b can be calculated by

$$\begin{gathered}
[(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T)\mathbf{K} + m\lambda\mathbf{I}]\mathbf{c} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T)\mathbf{y} \\
b = \bar{y} - \frac{1}{m}\mathbf{1}_m^T\mathbf{K}\mathbf{c}
\end{gathered} \tag{3.3}$$

where $\mathbf{K}$ is the kernel matrix so that $\mathbf{K}(i,j) = \mathrm{K}(\mathbf{x}_i, \mathbf{x}_j)$ and $\bar{y}$ is the mean of the elements of $\mathbf{y}$; $\mathbf{I}$ is the identity matrix that is appropriately sized; $\mathbf{1}_m$ is m-length vector with all entries equal to one. It can be seen from (3.3) that the solution requires inversion of the kernel matrix $\mathbf{K}$ ($m \times m$). In the context of microarray, m is usually in the order of tens or hundrends. That makes it possible to solve (3.3) directly. However, a quadratic programming problem is needed to be solved for SVM:

$$\begin{gathered}
\min_{c \in \Re^m, \xi \in \Re^m} C\sum_{i=1}^{m} \xi_i + \frac{1}{2}\mathbf{c}^T\mathbf{K}\mathbf{c} \\
\text{subject to:} \quad y_i(\sum_{j=1}^{m} c_j \mathrm{K}(\mathbf{x}_i, \mathbf{x}_j) + b) \geq 1 - \xi_i,\ i = 1, 2, \ldots, m \\
\xi_i \geq 0,\ i = 1, 2, \ldots, m
\end{gathered} \tag{3.4}$$

where C is used as the regularization parameter insteand of λ and has the relationship: $C = 1/2m\lambda$ (Rifkin, 2002). He reported that RLSC and SVM can achieve equivalent classification performance; so the choice of the two classifiers should be based on computational tractability. It can also be seen in Table 3.3 that (3.3) can be more efficiently solved than (3.4) on the benchmark microarray datasets that are used.

In this chapter, a new classification method is proposed to combine dimension-reduction methods with RLSC. Two dimension reduction-methods, KPLS and KPCA were used and then a linear RLSC was followed. Using the mutually orthogonal property of components extracted, the RLSC can be derived to have a very simple solution. In Section 3.2.2 and Section 3.2.3, the algorithms of KPLS and KPCA are introduced first; then the RLSC is derived based on components extracted from the two dimension-reduction methods.

3.2.2 Kernel Partial Least Squares

Partial least squares (PLS) is a regression technique for modeling a linear relationship between a set of input samples $\{\mathbf{x}_i\}_{i=1}^m$ ($\mathbf{x}_i$ is $n \times 1$ vector for $i = 1,2,\ldots,m$) and a set of output responses $\{y_i\}_{i=1}^m$ (y_i is scalar for $i = 1,2,\ldots,m$) (Garthwaite, 1994). Further we assume centered input and output variables; i.e. the input and output variables have zero means across all samples. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]^T$. PLS decomposes $\mathbf{X}$ into the form:

$$\mathbf{X} = \mathbf{T}\mathbf{V}^T + \mathbf{R} \tag{3.5}$$

where $\mathbf{T}$ $(m \times k)$ and $\mathbf{V}$ $(n \times k)$ are matrices of k score vectors and k loading vectors, $\mathbf{R}$ $(m \times n)$ is the matrix of residuals. The score vectors can be found by maximizing the covariance between $\mathbf{X}$ and $\mathbf{y}$, i.e.,

$$\max_{\|\mathbf{r}\|=1}[\mathrm{cov}(\mathbf{Xr}, \mathbf{y})]^2 = [\mathrm{cov}(\mathbf{Xh}, g\mathbf{y})]^2 = [\mathrm{cov}(\mathbf{t}, \mathbf{l})]^2 \tag{3.6}$$

where $\mathbf{r}$ is a n-length vector with its norm constrained to be one, $\mathbf{h}$ is a n-length weight vector and g is a scalar, and $\mathbf{t}$ and $\mathbf{l}$ are both m-length score vectors for $\mathbf{X}$ and $\mathbf{y}$, respectively. Note that the covariance between the two m-length vectors $\mathbf{t} = [t_1, t_2, \ldots, t_m]^T$ and $\mathbf{l} = [l_1, l_2, \ldots, l_m]^T$ is defined as

$$\mathrm{cov}(\mathbf{t}, \mathbf{l}) = \left[\sum_{i=1}^{m} (t_i - \bar{t})(l_i - \bar{l}) \Big/ (m-1) \right] \tag{3.7}$$

where $\bar{t}$ and $\bar{l}$ are means of $t_1, t_2, \ldots, t_m$ and $l_1, l_2, \ldots, l_m$, respectively. After obtaining one pair of score vectors $\mathbf{t}$ and $\mathbf{l}$, $\mathbf{X}$ and $\mathbf{y}$ are deflated by $\mathbf{t}$ and the procedure can be repeated to obtain a new pair of $\mathbf{t}$ and $\mathbf{u}$. This can be iterated for k times where k is the number of components. The different forms of deflation correspond to different variants of PLS (Wegelin, 2000). The SIMPLS (de Jong, 1993) algorithm is chosen, which provides the same solution as PLS1 (Wold, 1966) in the case of one-dimensional output. The score vectors for $\mathbf{X}$ produced by SIMPLS are mutually orthogonal.

Now consider a nonlinear transformation of the input samples $\{\mathbf{x}_i\}_{i=1}^m$ from the original input space $\Re$ into RKHS H; i.e. mapping $\Phi : \mathbf{x}_i \in \Re^n \to \Phi(\mathbf{x}_i) \in H$. The goal of the KPLS method is to construct a linear PLS model in the nonlinear RKHS H. Therefore, a nonlinear KPLS is effectively obtained and the mutual orthogonality of components could be retained. Denote $\mathbf{\Phi}$ as a $m \times n_0$ matrix of input samples in the feature space, whose i th row is the vector $\Phi(\mathbf{x}_i)^T$ where n_0 is the dimension of $\Phi(\mathbf{x}_i)$ and it could be infinite (e.g. if the RBF kernel is used).

The linear SIMPLS method to obtain the component vectors can be written in the RKHS as

$$\mathbf{t} = \mathbf{\Phi}\mathbf{\Phi}^T\mathbf{y}, \quad \|\mathbf{t}\| \to 1 \tag{3.8}$$

$$\mathbf{l} = \mathbf{y}(\mathbf{y}^T\mathbf{t}) \tag{3.9}$$

$\mathbf{t}$ is also called the PLS component. Here the symbol "$\to$" denotes normalization to one. Then the deflation rule is

$$\mathbf{\Phi} \leftarrow \mathbf{\Phi} - \mathbf{t}(\mathbf{t}^T\mathbf{\Phi}) \tag{3.10}$$

$$\mathbf{y} \leftarrow \mathbf{y} - \mathbf{t}(\mathbf{t}^T\mathbf{y}) \tag{3.11}$$

where the symbol "$\leftarrow$" denotes assignment of the value. Denote the sequence of consecutive $\mathbf{t}$ and $\mathbf{l}$ obtained as m-length vectors, $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k$ and m-length vectors, $\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_k$, and let $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k]$ and $\mathbf{L} = [\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_k]$. Instead of explicitly mapping the input data, the so called "kernel trick" is used resulting in

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T \tag{3.12}$$

where $\mathbf{K}$ represents the $m \times m$ kernel Gram matrix of the dot products between all samples $\{\Phi(\mathbf{x}_i)\}_{i=1}^m$ in the feature space; i.e.

$$\mathbf{K}(i,j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = \mathrm{K}_{\mathrm{ker}}(\mathbf{x}_i, \mathbf{x}_j) \tag{3.13}$$

where $\mathbf{K}(i,j)$ denotes the jth element of the ith row of $\mathbf{K}$ and $\mathrm{K}_{\mathrm{ker}}$ is a selected kernel function. $\mathbf{K}$ is now used in the deflation in place of $\mathbf{\Phi}$ as follows (Rosipal and Trejo, 2001):

$$\mathbf{K} \leftarrow (\mathbf{I} - \mathbf{t}\mathbf{t}^T)\mathbf{K}(\mathbf{I} - \mathbf{t}\mathbf{t}^T) \tag{3.14}$$

(3.10) is now replaced with (3.14). Therefore, the deflated kernel Gram matrix is obtained by the original kernel matrix and the PLS component. The assumption of zero means of the variables of $\mathbf{X}$ in linear PLS should also be held in KPLS. To centralize the mapped data in H, the following equation must be applied (Schölkopf *et al.*, 1998):

$$\mathbf{K} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T)\mathbf{K}(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T) \tag{3.15}$$

Assuming we have a set of test samples $\{\mathbf{z}_i\}_{i=1}^{m_t}$ ($\mathbf{z}_i$ is a n-length vector for $i = 1, 2, \ldots, m_t$), the projection of test samples into the RKHS (Rosipal and Trejo, 2001) is

$$\mathbf{T}_t = \mathbf{K}_t\mathbf{L}(\mathbf{T}^T\mathbf{K}\mathbf{L})^{-1} \tag{3.16}$$

where $\mathbf{T}_t$ is a $m_t \times k$ matrix with the columns representing the k KPLS components and the rows representing the m_t test samples in the reduced dimensional space. $\mathbf{K}_t$ is the $m_t \times m$ test set kernel Gram matrix so that

$$\mathbf{K}_t(i,j) = \mathrm{K}_{\ker}(\mathbf{z}_i, \mathbf{x}_j) \tag{3.17}$$

$\mathbf{T}^T\mathbf{KU}$ is an upper triangular matrix and thus invertible (Rosipal and Trejo, 2001). $\mathbf{K}_t$ should also be centered as

$$\mathbf{K}_t = (\mathbf{K}_t - \frac{1}{m}\mathbf{1}_{m_t}\mathbf{1}_m^T\mathbf{K})(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T) \tag{3.18}$$

3.2.3 Kernel Principal Components Analysis

PCA can be implemented by singular value decomposition (SVD) (Golub, 1996). Firstly, the linear SVD is introduced. According to the definition of SVD, $\mathbf{X}$ is decomposed as

$$\mathbf{X} = \mathbf{USV}^T \tag{3.19}$$

where the columns of $\mathbf{U}$ ($m \times m$) and $\mathbf{V}$ ($n \times m$) are called left and right singular vectors, respectively; $\mathbf{S}$ ($m \times m$) is a diagonal matrix with its entries being singular values. Using the terminology of PLS, here $\mathbf{T} = \mathbf{U}$ are score vectors[1] and $\mathbf{VS}$ are loading vectors. The classical solution for SVD is to compute $\mathbf{T}$ as the eigenvectors of $\mathbf{XX}^T$ because

$$\mathbf{XX}^T\mathbf{T} = \mathbf{TS}^2 \tag{3.20}$$

[1] Note that in Chapter 2, score vectors are defined as $\mathbf{US}$. The difference relies in whether score vectors are scaled by singular values or not. This is for convenience of equation derivation and will not cause any difference in final results.

where $\mathbf{S}^2$ is a diagonal matrix with its elements being the eigenvalues and also the squares of the singular values. Applying the mapping $\Phi : \mathbf{x}_i \in \Re^n \rightarrow \Phi(\mathbf{x}_i) \in \mathrm{H}$, the kernel Gram matrix $\mathbf{K}$ ($m \times m$) can be used to replace $\mathbf{X}\mathbf{X}^T$ in (3.20). The KPCA calculation becomes

$$\mathbf{KT} = \mathbf{TS}^2 \tag{3.21}$$

and the projection of test samples is

$$\mathbf{T}_t = \mathbf{K}_t \mathbf{TS}^{-2} \tag{3.22}$$

It should be noticed that in both KPLS and KPCA, all computations are performed around the kernel matrix $\mathbf{K}$. In the context of microarray data, since the size of samples, m, is very small, the computation of KPLS and KPCA is small too.

3.2.4 Regularized Least Squares Classification

Now we can formulate the linear RLSC classification method using the components produced by KPLS or KPCA. Given $\mathbf{x}$ the input sample, a linear classification function is

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{3.23}$$

where $\mathbf{w} = [w_1, w_2, \ldots, w_k]^T$ is the weight vector and b is the bias. A sample $\mathbf{x}$ is classified as 1 or -1 based on whether $f > 0$ or $f < 0$. $f = 0$ gives the separating hyperplane. Denote by $\mathbf{T}$ a $m \times k$ matrix of the projection of training samples using KPLS or KPCA so that $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k]$. Each row of the $\mathbf{T}$ is a input sample while each column of the $\mathbf{T}$ is a KPLS or KPCA component. The RLSC seeks the classification function by solving the following minimization function:

$$\min_{\mathbf{w},b} J(\mathbf{w}, b) = \frac{1}{2m}\left\|\mathbf{b} + \mathbf{T} \cdot \mathbf{w} - \mathbf{y}\right\|^2 + \frac{1}{2}\lambda\left\|\mathbf{w}\right\|^2 \tag{3.24}$$

where J is the objective function and $\mathbf{b}$ is the m-length vector with all elements equal to b. Let $\mathbf{w}_1 = [b, \mathbf{w}]^T$, $\mathbf{w}_2 = [0, \mathbf{w}]^T$ and $\mathbf{T}_1 = [\mathbf{1}_m, \mathbf{T}]$. (3.24) can equivalently be written as

$$\min_{\mathbf{w}_1} J(\mathbf{w}_1) = \frac{1}{2m}\|\mathbf{T}_1 \cdot \mathbf{w}_1 - \mathbf{y}\|^2 + \frac{1}{2}\lambda\|\mathbf{w}_2\|^2 \tag{3.25}$$

$\mathbf{w}$ and b can thus be simultaneously solved by (Appendix B)

$$\mathbf{w}_1 = \mathbf{P}^{-1}\mathbf{T}_1^T\mathbf{y} \tag{3.26}$$

with

$$\mathbf{P} = \begin{bmatrix} m & \sigma_1 & \cdots & \sigma_k \\ \sigma_1 & 1+\lambda & & \\ \vdots & & \ddots & \\ \sigma_k & & & 1+\lambda \end{bmatrix} \tag{3.27}$$

where t_{ij} is the jth element of $\mathbf{t}_i$ which is the score vector, and $\sigma_i = \sum_{j=1}^{m} t_{ij}$. It can be easily proved that $|\sigma_i| \le 1/2$. Thus $\mathbf{P}$ is symmetric and positive definite. $\mathbf{P}$ is also sparse with a dimension of only $(k+1)\times(k+1)$. Because $k < m \ll n$, it is obvious that this matrix can be inverted with a very small computational effort.

3.2.5 Components and Variance

There is no theoretical proof of how to select an optimal subset of components so that the classification performance can be maximized. The rule of thumb (Nguyen and Rocke, 2002) is to select 5 or 10 components for classification. This would usually give satisfactory results. In this chapter, the variance of variables captured by the components is proposed as a measure for components selection. A component which captures a large proportion of variance is always assumed to be useful for classification. For KPLS, there are two kinds of variance: the variance of label (VOL), i.e. the variance of output response (class label); and the variance of input variables

(VOI). Assuming $\mathbf{y}_k$ to be the output response variable after k KPLS components have been extracted. The proportion of the VOL captured by k components is

$$r_l = 1 - \frac{\|\mathbf{y}_k\|^2}{\|\mathbf{y}\|^2} \tag{3.28}$$

and the proportion of the VOI is

$$r_v = 1 - \frac{\text{trace}(\mathbf{K}_k)}{\text{trace}(\mathbf{K})} \tag{3.29}$$

where trace is the sum of the diagonal elements of matrix and $\mathbf{K}_k$ is the kernel matrix after k iterations. For KPCA, only VOI exists. Assuming the singular values of $\mathbf{S}$ is sorted in descending order, the proportion of the VOI is

$$r_v = \sum_{i=1}^{k} s_i^2 \Big/ \sum_{i=1}^{m} s_i^2 \tag{3.30}$$

The number k of components can be determined when a enough proportion of variance has been captured and the rest components can thus be considered as noise and discarded. Note that in the following, VOL and VOI are used to refer to both the variance and the proportion of the variance captured by the components.

3.3 Results

3.3.1 Datasets and Experimental Setup

Seven publicly available cancer datasets were chosen from Li and Liu (2002). They are: breast cancer (BREAST), central nervous systems tumor (CNS), colon tumor (COLON), acute leukemia cancer (LEUKEMIA), lung cancer (LUNG), ovarian cancer (OVARIAN) and prostate cancer (PROSTATE). A brief description of these datasets is listed in Table 3.1. All gene expression profiles (proteomic patterns) were log-transformed and normalized to have zero mean and unit variance. Because the number of genes in a microarray dataset is very large, T-statistics is used to filter out those genes that are not differentially expressed between the two classes. The confidence level is set to $\alpha = 0.05$. The number of genes after filtering is also given in Table 3.1.

Two kernels were used: linear and RBF. The RBF kernel is defined as

$$\mathrm{K}_{\mathrm{ker}}(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (n\omega)) \tag{3.31}$$

where n is the number of variables and ω is chosen from $[1, 10^2, 10^4, 10^6]$. The regularization parameters λ for RLSC and C for SVM were both chosen from $[0.0001, 0.01, 1, 100, 10000]$. The optimal kernel and parameters were determined by three-fold cross-validation (CV) (Duda *et al.*, 2000) so that the minimum CV errors were obtained. When there are several sets of parameters yielding the minimum CV error, the simplest model is always preferred: a larger λ or a smaller C is always chosen; a linear kernel is always preferred to an RBF kernel; a larger ω is always preferred to a smaller ω.

All codes of KPLS, KPCA, and RLSC were written in MATLAB®. An SVM toolbox written by Gunn (2001) was used. The SVM toolbox was written in MATLAB® codes except the quadratic programming solver was written in C and was compiled into an executable module. All experiments were carried out on a MATLAB® version 6.5 under UNIX environment.

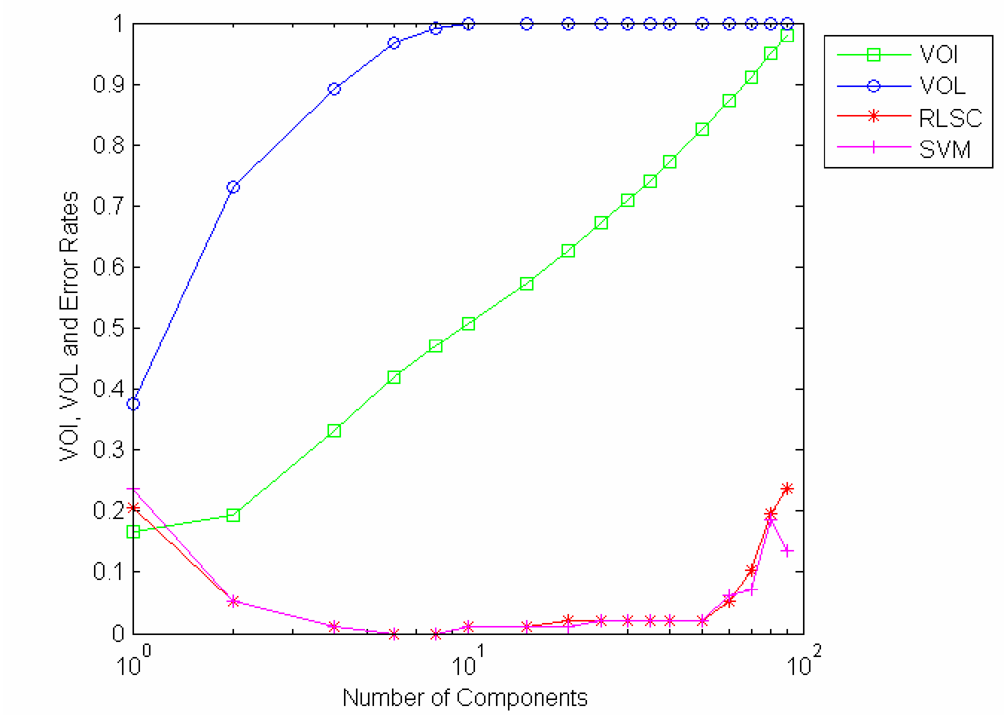
Table 3.1 Description of the datasets.

Dataset	Genes	Selected Genes	Samples
Breast Cancer	24481	3600	97
Central Nervous Systems	7129	992	60
Colon Tumor	2000	555	62
Acute Leukemia	7129	2683	72
Lung Cancer	12533	6005	181
Ovarian Cancer	15154	9115	253
Prostate Cancer	12600	6783	102

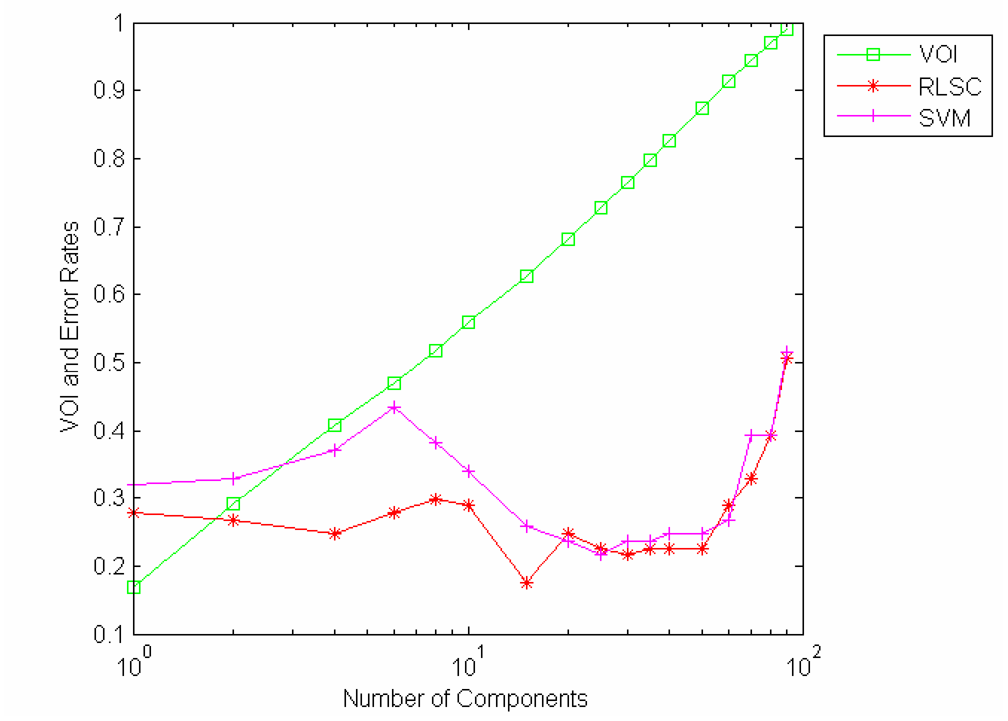
3.3.2 Components Selection

To determine how the VOL and the VOI correlate with the classification performance, we vary the number of components and observe the 10-fold CV error rates on the seven datasets. Linear kernel is used for both RLSC and SVM with $\lambda = 0.0001$ and $C = 1$ fixed. The results are summarized in Figure 3.1a-g.

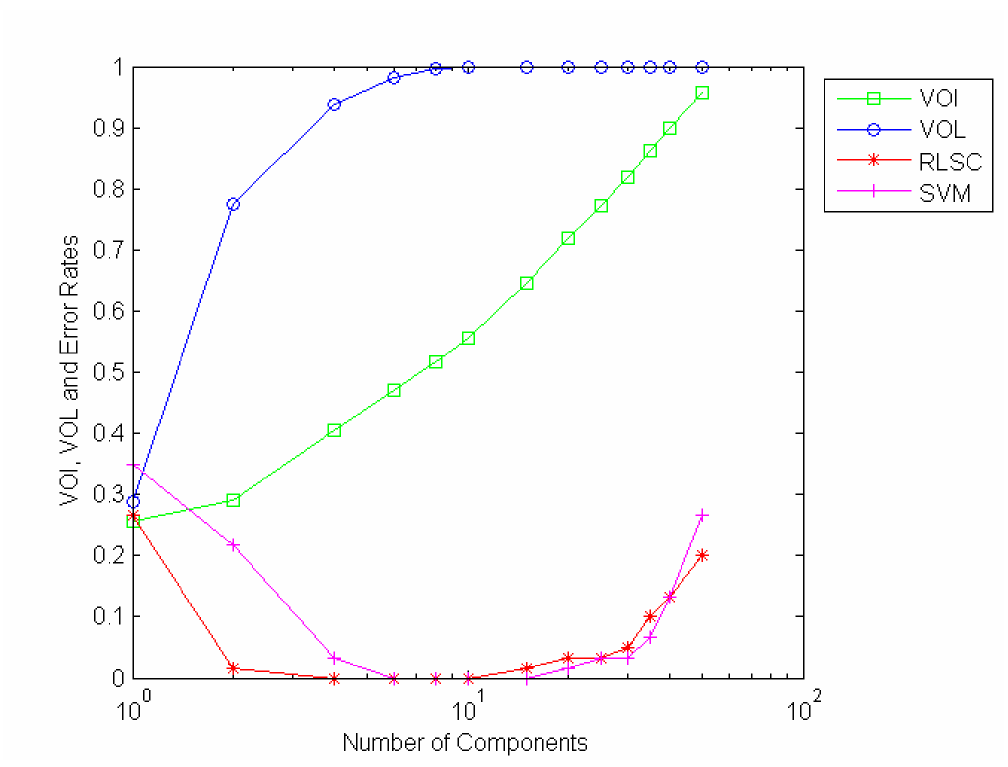
It can be observed that the VOL increases rapidly with the number of KPLS components. Usually the VOL can rearch almost 1.0 within 10 components. This indicates the importance of the first a few KPLS components. The CV errors show a strong correlation with the VOL. They usually reach the minimum when the VOL increases to 1.0. However, the VOI for KPLS components is not very indicative. The VOI may still be less than 0.5 when the VOL approaches 1.0 and the CV error reaches minimum. So in the following experiments, we select KPLS components when the VOL reaches 0.99 without regarding to the VOI. For KPCA, the VOI shows strong correlation with the CV error. The CV error decreases with the VOI increases. The minimum of CV error can be obtained when the VOI is in the range of $[0.8,\ 0.9]$. However, if more KPCA components are included, the CV error increases. Therefore, KPCA components are chosen based on the VOI at 0.85 in Section 3.3.3 and Section 3.3.4.


VOI
VOL
RLSC
SVM
VOI, VOL and Error Rates
Number of Components
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2

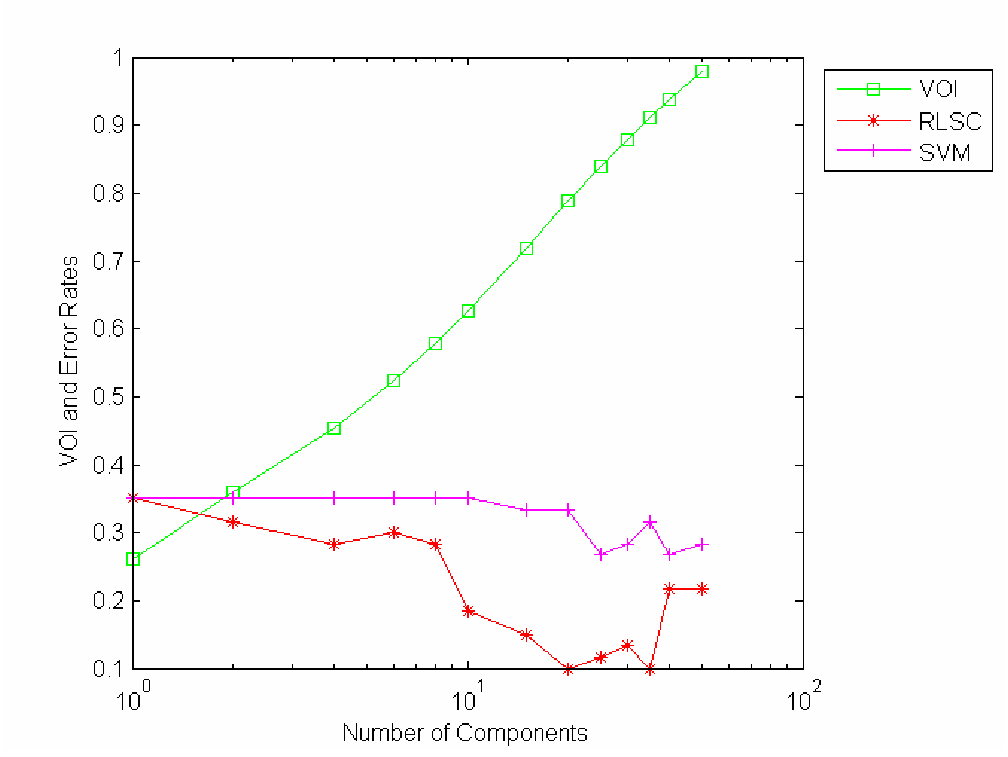
(a)(i)


VOI
RLSC
SVM
VOI and Error Rates
Number of Components
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2

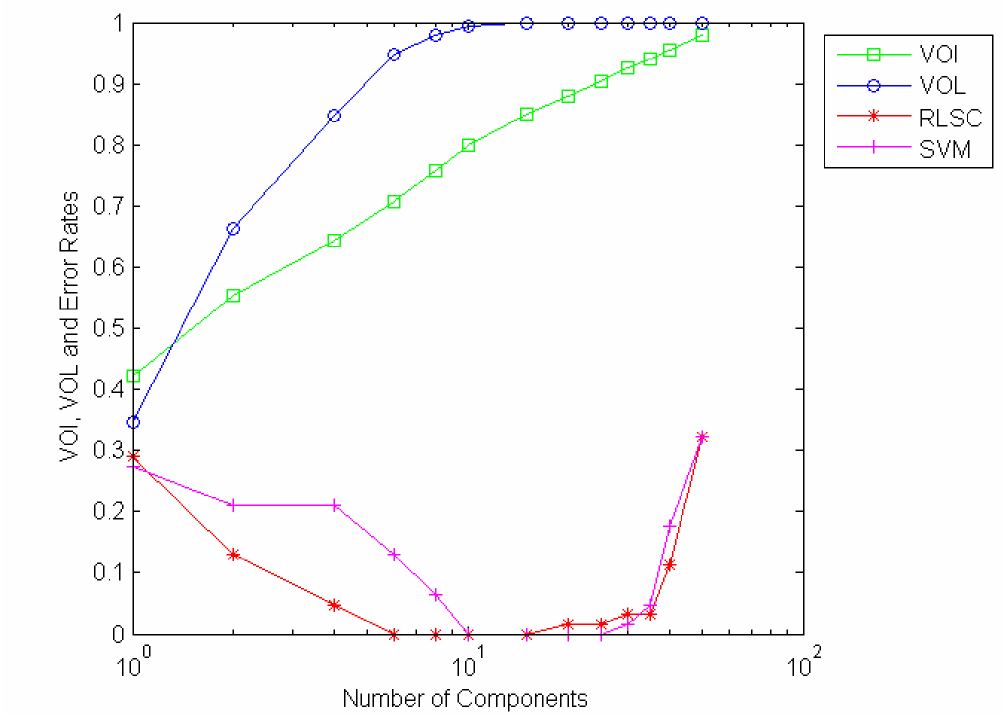
(a)(ii)


VOI
VOL
RLSC
SVM
VOI, VOL and Error Rates
Number of Components
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2

(b)(i)

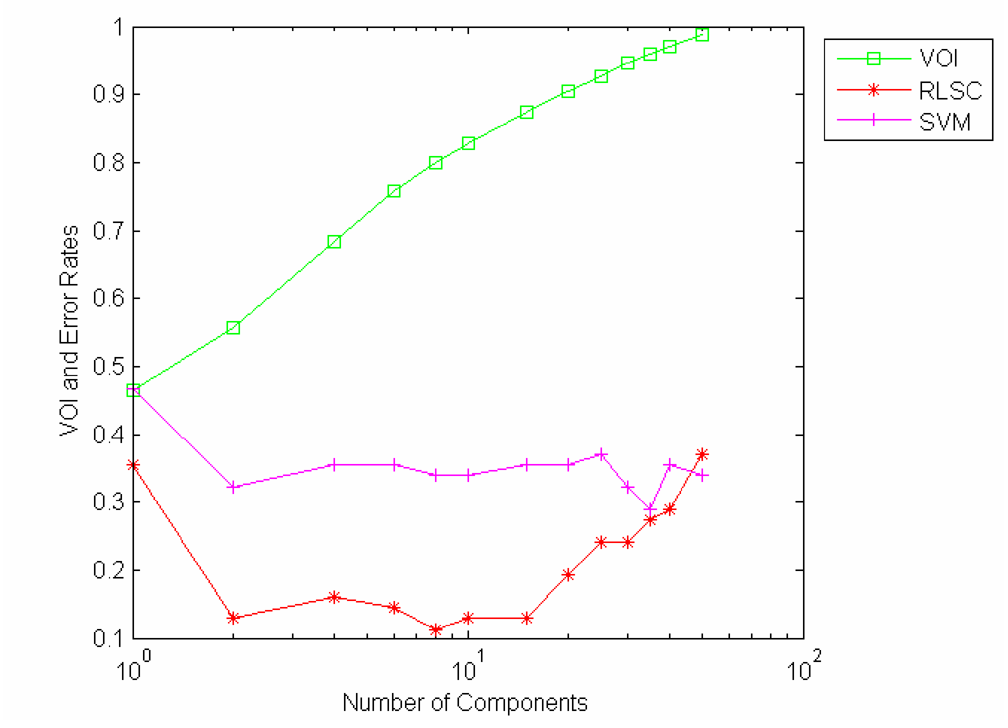

VOI
RLSC
SVM
VOI and Error Rates
Number of Components
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2

(b)(ii)



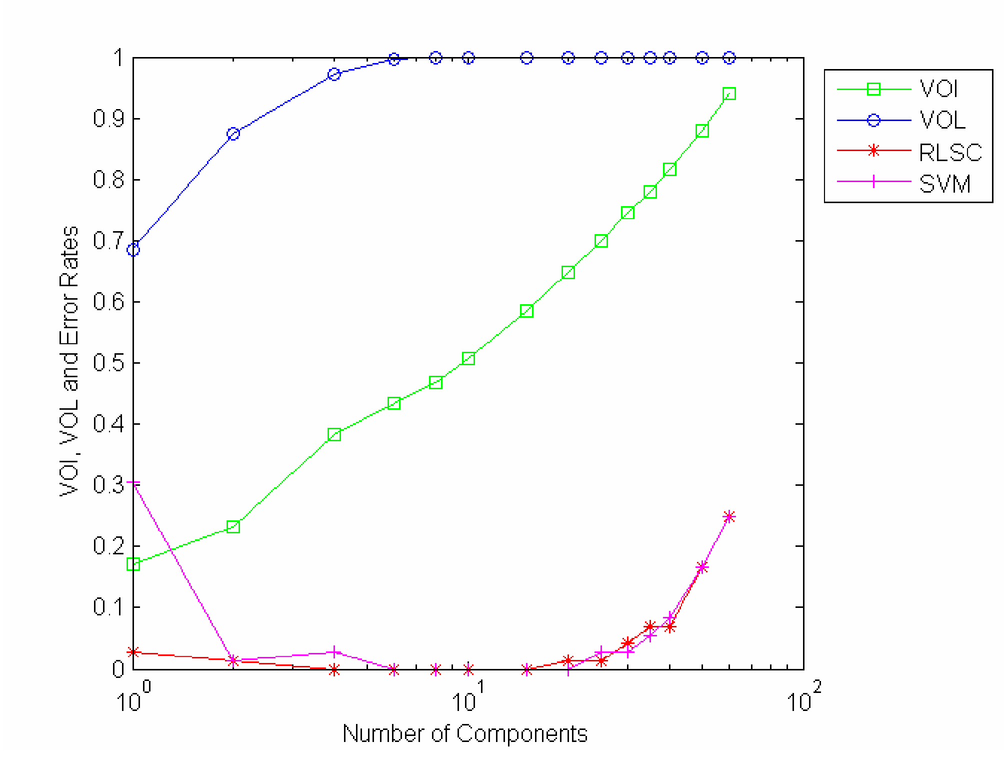
VOI, VOL and Error Rates
Number of Components
VOI
VOL
RLSC
SVM

(c)(i)

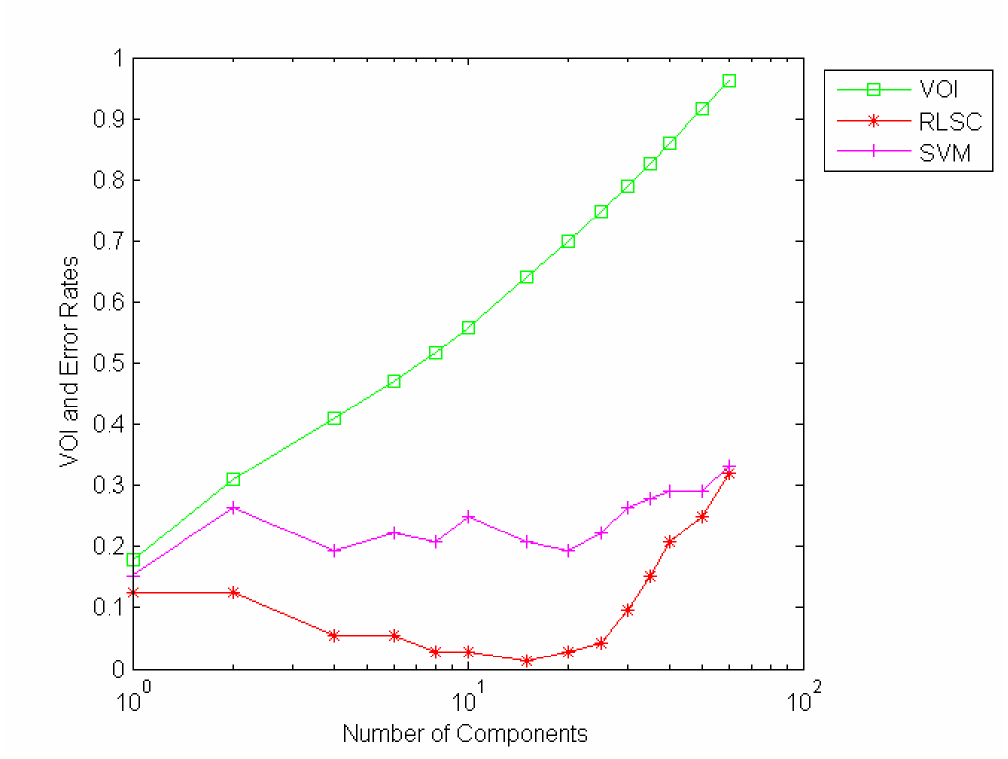


VOI and Error Rates
Number of Components
VOI
RLSC
SVM

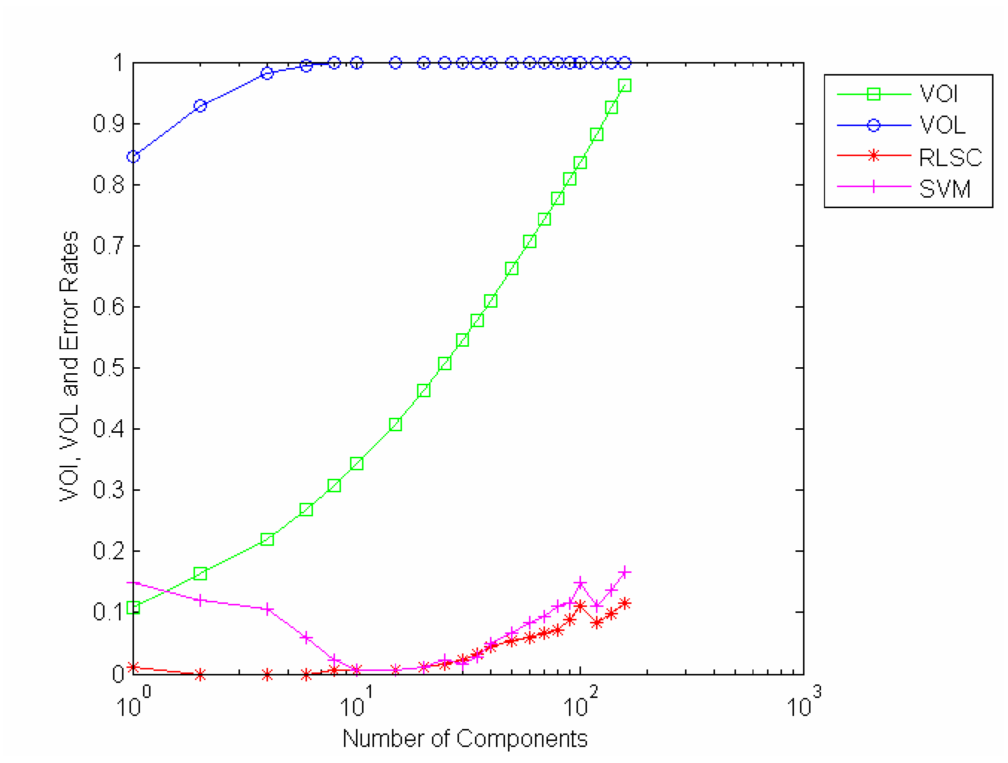
(c)(ii)


VOI
VOL
RLSC
SVM
VOI, VOL and Error Rates
Number of Components
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2

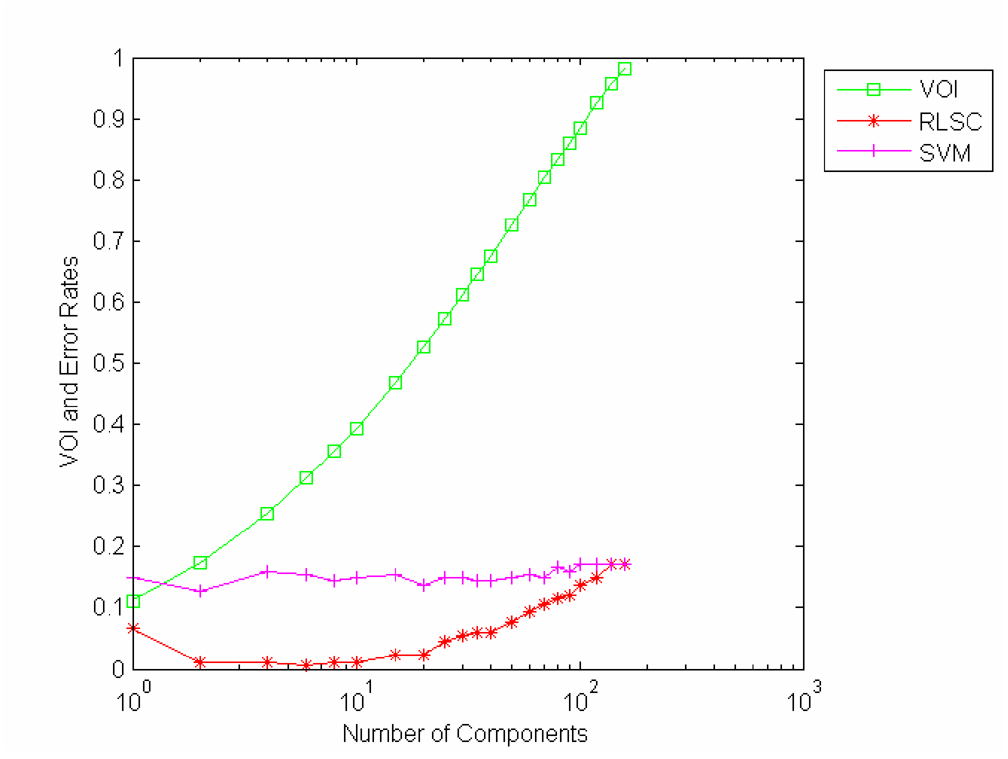
(d)(i)


VOI
RLSC
SVM
VOI and Error Rates
Number of Components
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2

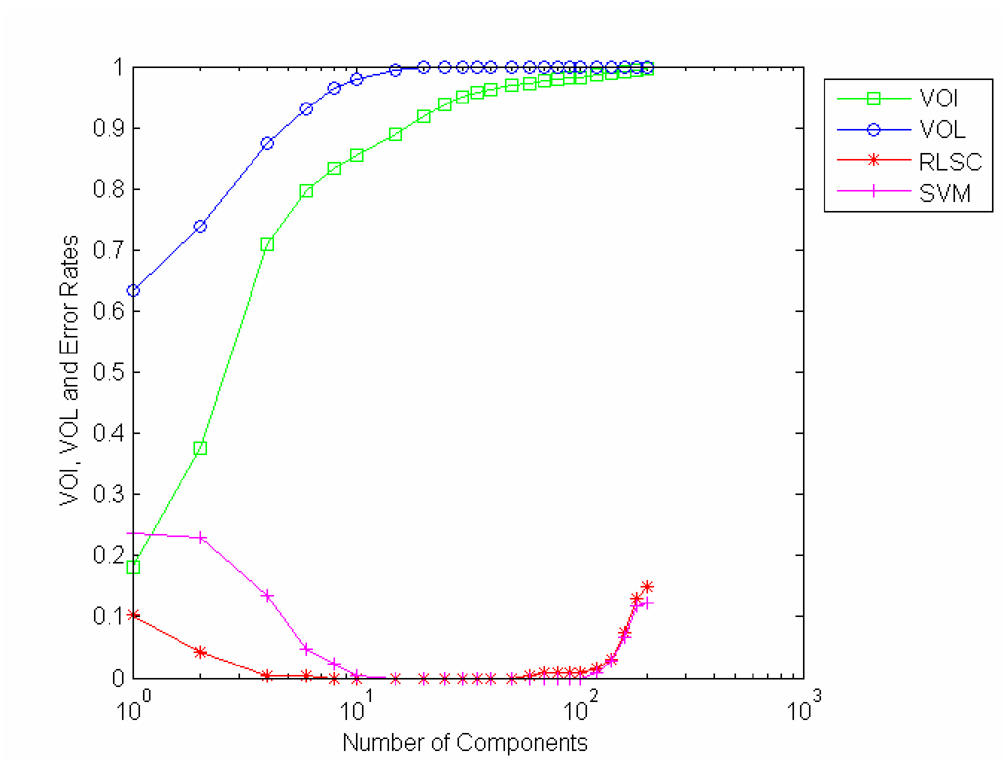
(d)(ii)


VOI
VOL
RLSC
SVM
VOI, VOL and Error Rates
Number of Components
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2
10^3

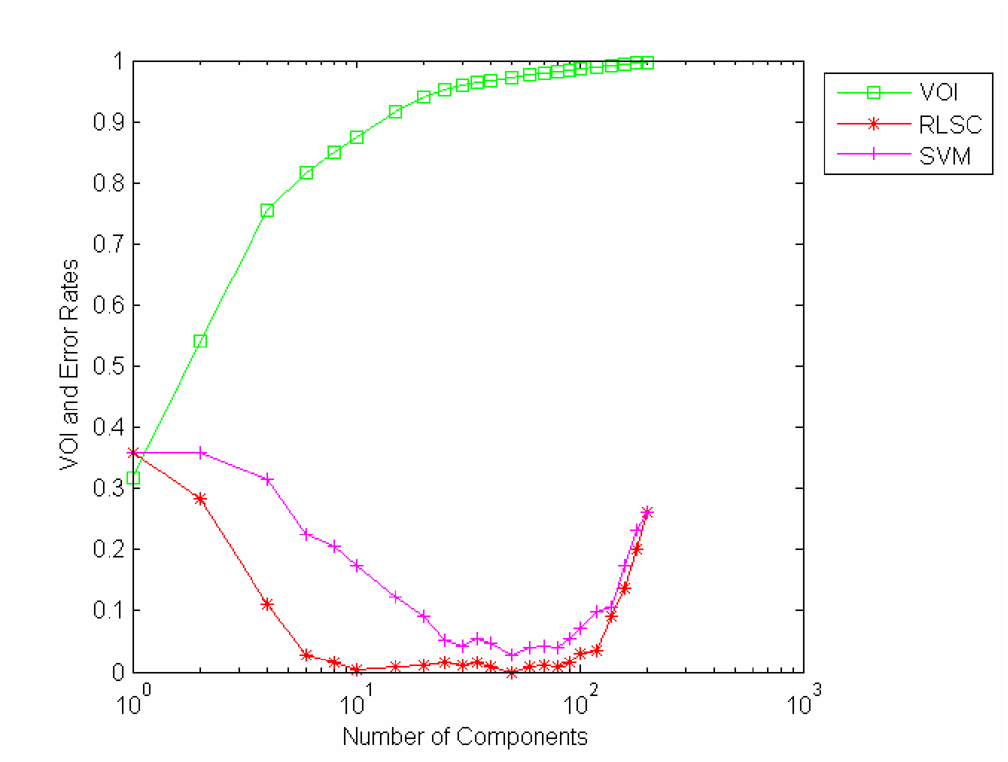
(e)(i)


VOI
RLSC
SVM
VOI and Error Rates
Number of Components
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2
10^3

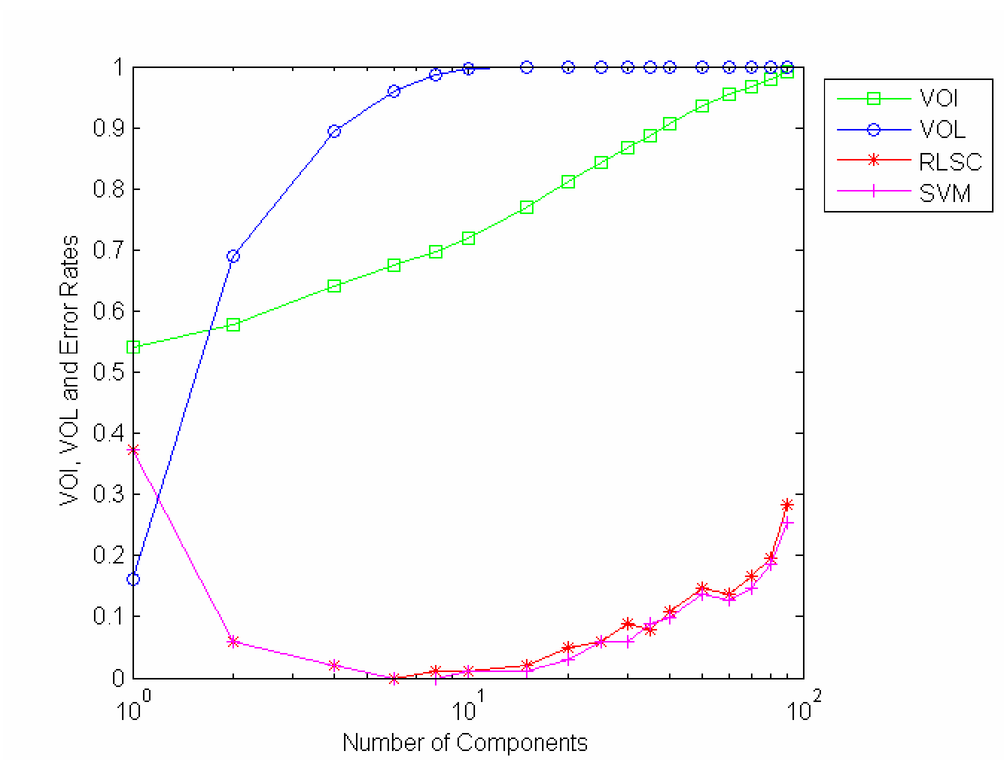
(e)(ii)


VOI
VOL
RLSC
SVM
VOI, VOL and Error Rates
Number of Components
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2
10^3

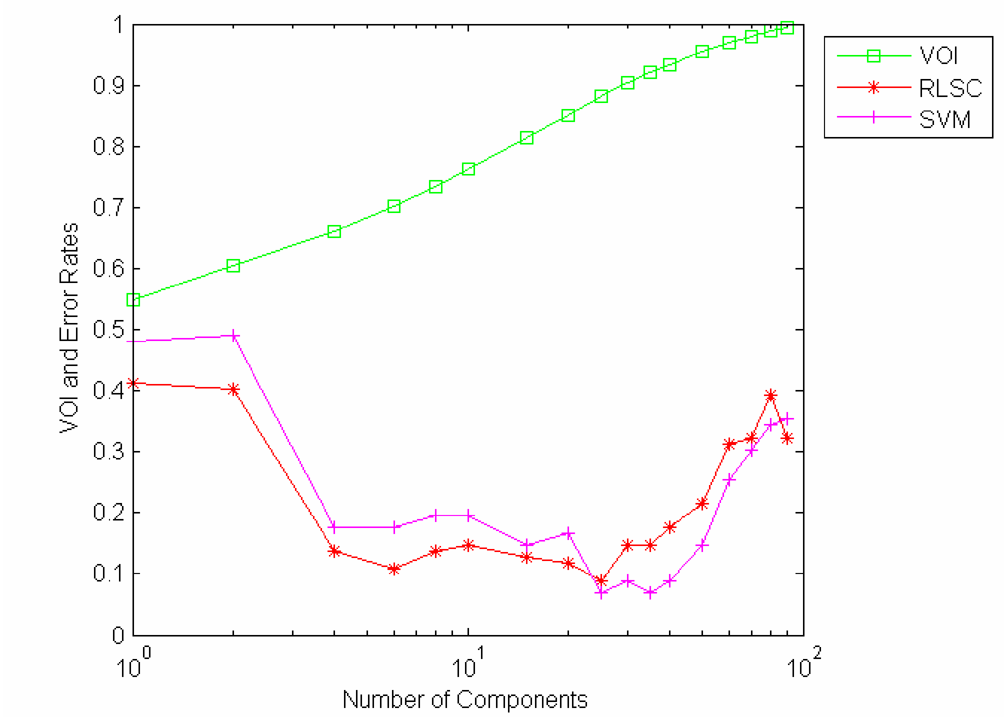
(f)(i)


VOI
RLSC
SVM
VOI and Error Rates
Number of Components
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^0
10^1
10^2
10^3

(f)(ii)



(g)(i)



(g)(ii)

Figure 3.1 VOL and VOI, 10-fold CV error rates vs. the number of components. KPLS and KPCA for upper and lower panels, respectively. (a) (i) and (ii) BREAST. (b) (i) and (ii) CNS. (c) (i) and (ii) COLON. (d) (i) and (ii) LEUKEMIA. (e) (i) and (ii) LUNG. (f) (i) and (ii) OVARIAN. (g) (i) and (ii) PROSTATE.

3.3.3 Classification Accuracies

One difficulty in assessing the accuracies of different classifiers is the small sizes of microarray samples. To avoid the variance that may occur due to a particular partition of the training and testing datasets, re-sampling method was used. The original dataset was randomly partitioned into a training dataset and a testing dataset according to a 2:1 ratio. The optimal parameters were determined by three-fold CV on training data and then the trained classifier was tested on the testing dataset. This process was repeated for one hundred times. Finally, the mean of testing error rates, μ, and its standard deviation, $\bar{\sigma}$, can be calculated as

$$\bar{\sigma} = \frac{\sigma}{\sqrt{l}} \tag{3.32}$$

where σ is the standard deviation of error rates and l is the number of random partitions.

There are two categories of classifers based on the classification algorithms: RLSC and SVM. Combining with the KPLS and KPCA, or without dimension reduction, we have six classifiers in total. The error rates along with the components used for the six classifiers are given in

Table 3.2. Because the components extraction is independent of classification, only the components used for RLSC are listed. KPLS+RLSC appears to be the best classifier among the six. Although KPCA+RLSC achieves excellent performance, it is statistically worse than KPLS+RLSC on BREAST, CNS and OVARIAN according to a T-statistics with 95% assurance. This shows the superiority of KPLS as dimension-reduction method for supervised classification. This is confirmed by the fact that KPLS+SVM outperforms KPCA+SVM. It is calculated that although KPLS+RLSC is not the best classifier on the LEUKEMIA and LUNG, it is not statistically worse than

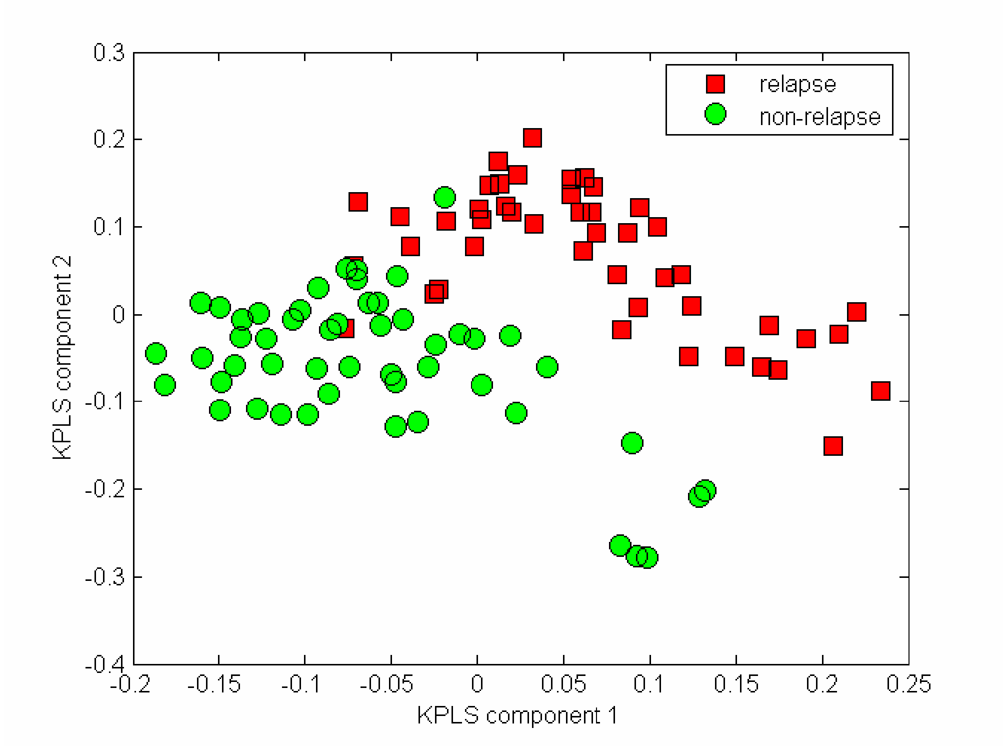
the best classifier based on a T-statistics (95%). RLSC and KPLS+SVM perform better than SVM and are the two closest classifiers to KPLS+RLSC.

Considerably less KPLS components are used than KPCA to construct RLSC classifiers. This means that the KPLS components are more related to class distinction. It is observed that the standard deviation of the mean KPLS component is much less than the counterpart of KPCA. This shows the stability of the number of the KPLS components used to construct classifiers. For example, on LEUKEMIA, the expected number of KPLS components is in between $[4.90, 4.98]$ with a 95% confidence according to a normal distribution, which is very close to 5.

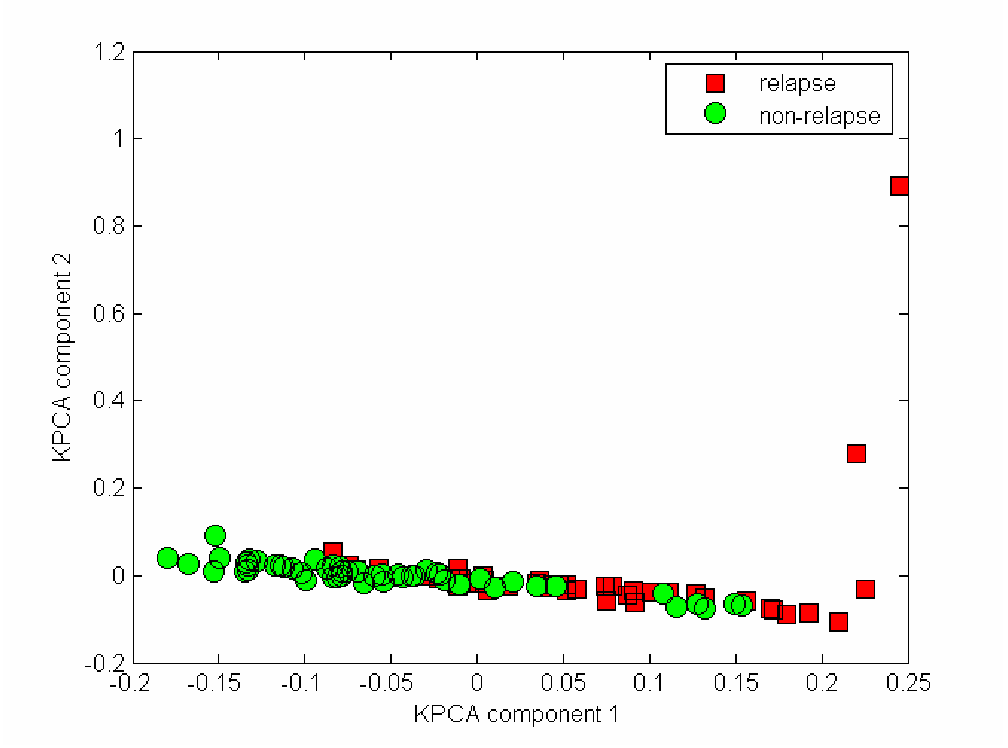
Among the seven cancer datasets, BREAST appears to be the most difficult to separate, while LUNG and OVARIAN are the easiest to separate. It is interesting to see the cluster plots using the two most significant KPLS and KPCA components for the above datasets. BREAST and LUNG are chosen (Figure 3.2). An RBF kernel with

Table 3.2 Testing error rates and components used of the six classifiers.

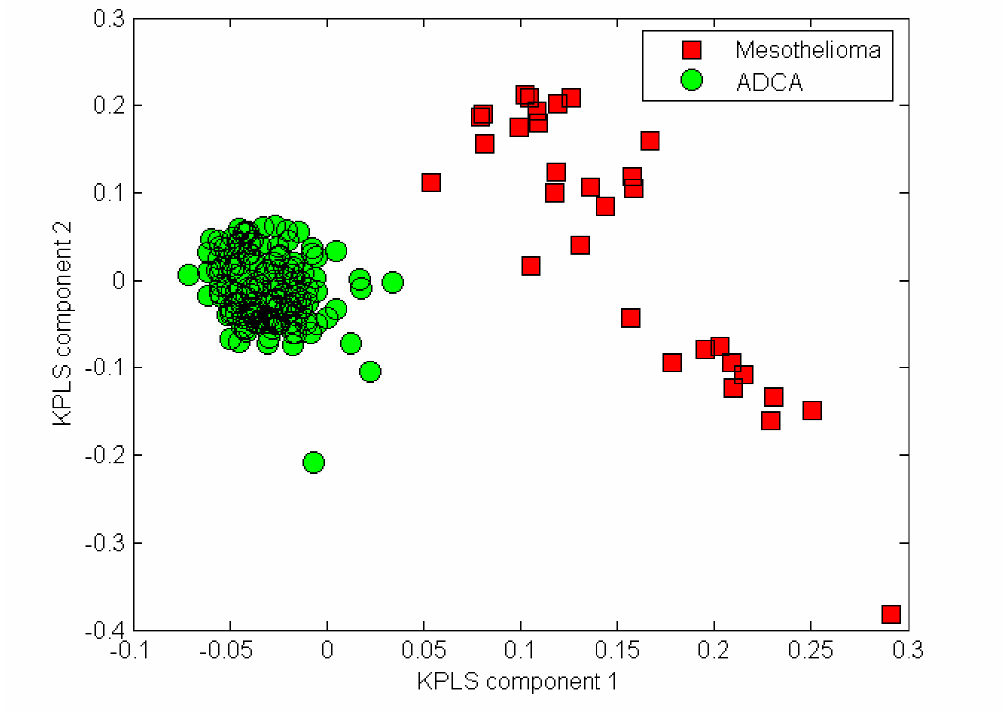
Dataset	Testing error rates (%) ($\mu, \bar{\sigma}$)			Components($\mu, \bar{\sigma}$)	
	KPLS+RLSC	KPCA+RLSC	RLSC	KPLS	KPCA
BREAST	**21.06,0.74**	23.94,0.65	23.09,0.66	6.48,0.08	33.08,0.76
CNS	**12.00,0.70**	16.80,0.73	13.50,0.78	6.27,0.06	20.03,0.33
COLON	15.30,0.55	**11.55,0.61**	13.10,0.61	7.48,0.13	10.71,0.33
LEUKEMIA	1.83,0.24	**1.46,0.22**	1.54,0.22	4.94,0.02	26.70,0.29
LUNG	0.58,0.08	0.53,0.08	**0.52,0.08**	5.05,0.04	58.39,0.98
OVARIAN	**0.08,0.04**	1.77,0.12	0.21,0.07	11.97,0.02	23.89,2.78
PROSTATE	**5.79,0.34**	10.85,0.52	5.88,0.30	7.87,0.06	25.45,1.05
	KPLS+SVM	KPCA+SVM	SVM	KPLS	KPCA
BREAST	22.50,0.69	24.91,0.68	21.53,0.73	--	--
CNS	12.50,0.75	16.95,0.92	14.85,0.81	--	--
COLON	14.75,0.65	17.90,0.74	27.85,1.97	--	--
LEUKEMIA	1.79,0.25	2.13,0.25	1.79,0.22	--	--
LUNG	0.78,0.10	0.75,0.10	0.55,0.08	--	--
OVARIAN	0.13,0.04	2.12,0.16	0.44,0.08	--	--
PROSTATE	6.47,0.31	11.29,0.53	6.62,0.38	--	--


relapse
non-relapse
KPLS component 2
KPLS component 1
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.2
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2
0.25

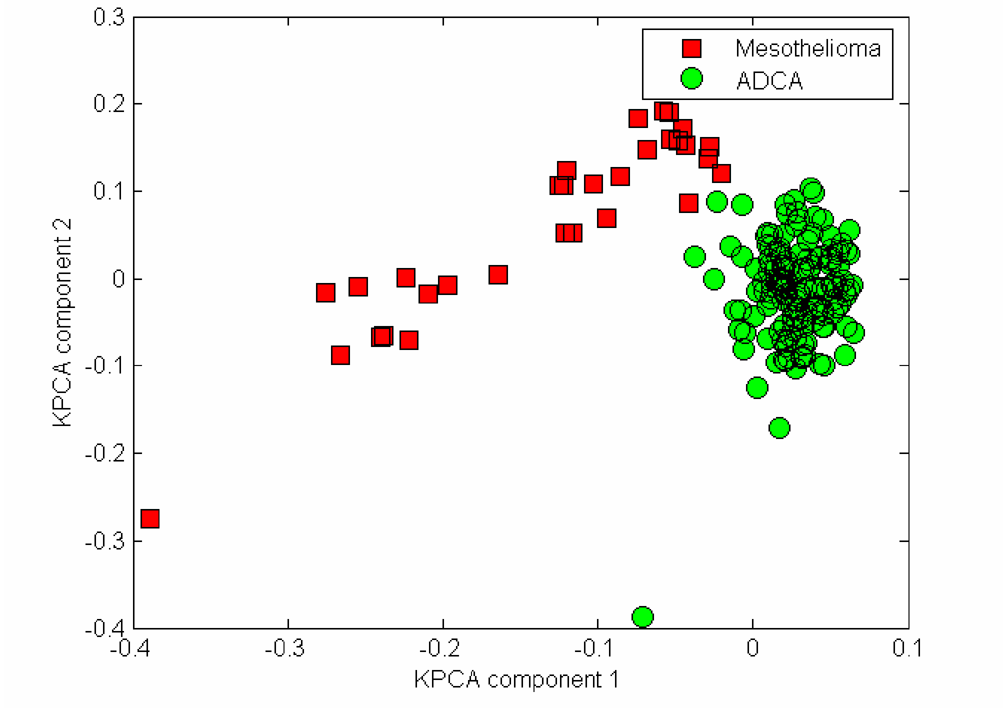
(a)


relapse
non-relapse
KPCA component 2
KPCA component 1
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.2
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2
0.25

(b)



(c)



(d)

Figure 3.2 Cluster plot of BREAST and LUNG using KPLS and KPCA components. (a) BREAST with KPLS. (b) BREAST with KPCA. (c) LUNG with KPLS. (d) LUNG

with KPCA. $\omega = 100$ was used. On BREAST, KPLS separates the two "relapse" and "non-relapse" classes with a few overlaps, while KPCA totally fails to produce separation, resulting in two cramped clusters. On LUNG, both component extraction methods successfully separate the two classes. KPLS produces a clear separation between the two classes. However, the two classes seem to almost "touch" each other at the boundary created by KPCA.

3.3.4 Computational Speed

The total computational time for training and testing was recorded in Table 3.3. The timer returns the total CPU time used by MATLAB® excluding the time for data loading and gene filtering. In the implementation of SVM in this work, the whole kernel matrix has been cached in memory during training and testing. Since the range of kernel parameters and regularization parameters is the same for the six classifiers, the size of the grids used for CV parameter selection is also the same. Therefore, the comparison is fair for the six classifiers. It is seen that the three classifiers in the RLSC category, and the three classifiers in the SVM category respectively took almost the same time. However, the SVM category presents itself as the less efficient classifier, taking almost twice the time of the RLSC category on COLON and OVARIAN. It is thus inferred that after dimension reduction, both RLSC and SVM can be enhanced. Nevertheless, the extra step of dimension reduction also costs time, which makes the three classifiers in the same category approximately equal in speed performance.

Table 3.3 Computational time (s) for the six classifiers.

Dataset	KPLS+RLSC	KPCA+RLSC	RLSC	KPLS+SVM	KPCA+SVM	SVM
BREAST	1610	1590	1531	2258	2221	2208
CNS	436	461	361	722	702	640
COLON	289	288	227	576	576	509
LEUKEMIA	1052	1086	1018	1415	1485	1351
LUNG	3590	4029	3745	6515	6590	6680
OVARIAN	7985	8500	7585	14361	14491	14034
PROSTATE	3052	3107	2898	3823	3600	3733

3.4 Discussions and Summary

The problem of concern is the irrelevant genes and the noise that commonly exist in microarray data. KPLS and KPCA are combined with linear RLSC for cancer classification. It is proved that they can perform equally well with, or even better, than the kernalized RLSC or SVM. Owing to the reduced dimension and simplified matrix inversion, the new methods are very efficient even with the extra step of dimension reduction.

It may be best to select the number of components by CV but huge computation is expected. This problem is tackled by plotting CV error rates along with the proportion of variance captured by the components versus the number of components that have been used. It can be seen from the graphs that the error rates and the proportion of variance are highly correlated. Therefore, components selection based on component variance appears to be a simple but effective method. From the high accuracies that have been achieved by KPLS+RLSC and KPCA+RLSC, the effectiveness of this method is further validated. More extensions that combine kernel dimension-reduction methods with other classification methods may be considered in the future.

Chapter 4. Reducing Multiclass Cancer Classification to Binary by Output-Coding and SVM

4.1 Introduction

Applying machine learning techniques to microarray data for cancer classification can achieve rather promising results. Previous work has been mainly in the field of binary classification and good accuracy can be obtained (Cho and Won, 2003). Multiclass classification, on the other hand, is more difficult (Yeang *et al*., 2001; Li *et al*., 2004) but gaining research momentum in machine learning. One of the more successful directions is to use error correcting output codes (ECOC) with binary classifiers. It was first proposed by Dietterich and Bakiri (1995) and many variants of the method have since been reported, some of which are very encouraging.

Li *et al*. (2004) have discussed combining different feature selection methods and coding techniques with classifiers for multiclass cancer classification. However, in this communication, we describe a more generalized output-coding scheme which considers different coding strategies and decoding functions in one single framework. The Support Vector Machine (SVM) is chosen as the binary classifier and the effectiveness of the various combinations is verified. It is noticed that by simply extending the codeword length of one of the coding strategies, an improvement can be achieved on the GCM data (Ramaswamy *et al*., 2001). Our approach is compared with three other popular machine learning methods, the K-nearest neighbor (KNN), the C4.5 decision tree and the neural network (NN).

Since microarray data has the characteristics that the number of genes is much larger than the number of samples, feature selection is an important issue before classification. Thus, we also evaluate the three major categories of feature selection: gene ranking, dimension reduction and recursive feature elimination (RFE).

4.2 Methods

4.2.1 Output-Coding for Multiclass Classification

Assume that we have a set of m microarray samples: $(\mathbf{x}_i, y_i)$, $i = 1, 2, \ldots, m$, where $\mathbf{x}_i \in \Re^n$ is a vector of length n representing gene expression levels and $y_i \in \{1, 2, \ldots, k\}$ is the class label of the i th sample. In multiclass context, $k > 2$. The classification algorithm aims to find a mapping $M : \Re^n \to \{1, 2, \ldots, k\}$ using the m training samples. The output-coding method decomposes the k -class problem into a set of l binary subproblems, trains the resulting l base classifiers and then combines the l outputs to predict the class label. We have adopted the generalized scheme proposed by Allwein *et al*. (2000). It begins with a given coding matrix

$$\mathbf{M} \in \{-1, 0, +1\}^{k \times l}$$

for which each row $\mathbf{r}_i$ ($i = 1, 2, \ldots, k$) represents the codeword of the i th class and each column $\mathbf{s}_j$ ($j = 1, 2, \ldots, l$) represents the j th base classifier. Each row $\mathbf{r}_i$ must be unique for its corresponding class, i.e. $\forall i, j$ such that $i \neq j$, $1 \leq i$ and $j \leq k$, we have $\mathbf{r}_i \neq \mathbf{r}_j$. $\mathbf{M}(i, j) = 1$ or -1 means that the i th class should be considered as positive or negative for the j th base classifier, respectively. If $\mathbf{M}(i, j) = 0$, the i th class is simply ignored by the j th base classifier. Therefore, the j th column of $\mathbf{M}$ represents a partition of the original multiclass data into two classes as positive and negative. Any binary classifier can be used to solve the induced two-class problem, e.g. the SVM.

Let f_s ($s = 1, 2, \ldots, l$) denote the l base classification functions. Given a microarray sample $\mathbf{x}$, let $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_l(\mathbf{x}))$; then its class label y is predicted as

$$y = \arg\min_i d(\mathbf{r}_i, \mathbf{f}(\mathbf{x})) \tag{4.1}$$

where d is called the decoding function. Different coding matrices and decoding functions can have significant influence on the classification accuracy of the output-coding scheme. By adopting this generalized scheme, we can combine some of the researchers' work into one single system.

4.2.2 Coding Matrix

There are various methods to generate coding matrices. Different coding matrices may have substantial effect on classification accuracy. Probably the simplest approach is to set $\mathbf{M}$ as a $k \times k$ square matrix with all its diagonal elements 1 and all the other elements -1. Thus, it is equivalent to a binary problem for each of the k classes. That is, for base classifier i ($1 \le i \le k$), we train the classifier in which all samples of class i are considered as positive and all samples of the other classes are considered as negative. This is called the one-versus-all (OVA) approach.

Another approach by Hastie and Tibshirani (1998) is to use a binary classifier to distinguish one pair of classes at a time. Meanwhile, the other classes are simply ignored. So there are totally $\binom{k}{2}$ base classifiers to induce. If $k = 3$, the coding matrix is given by

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{bmatrix} \tag{4.2}$$

This is called the all-pairs (AP) approach.

Error correcting output codes (ECOC) was proposed by Dietterich and Bakiri (1995). They argued that if the minimum hamming distance between a pair of rows of the coding matrix is c, the output codes have the ability to correct $\lfloor (c-1)/2 \rfloor$ errors of the base classifiers. The OVA approach thus does not have correcting power because $c = 2$. There are many approaches to generate ECOC. Designing a good set of ECOC requires both row and column separation of the coding matrix. Two major coding strategies, the random coding and the exhaustive coding, are given as follows:

(A) Random coding: Let $l = \lceil 10\log_2(k) \rceil$ (Dietterich and Bakiri, 1995). Each element of the coding matrix is assigned a value from $\{-1, 1\}$ uniformly at random. Then, a hill-climbing procedure (Ricci and Aha, 1998) is applied. The procedure can usually improve the averaged and minimum hamming distances between pairs of rows of the coding matrix so that better classification accuracy can be achieved.

(B) Exhaustive coding: Firstly let $l = 2^{k-1}$. The columns of the coding-matrix are generated by enumerating all possible bit-strings with their length fixed to k. The complementary column is excluded. Because exactly one column is assigned +1 for all of its elements, it is deleted from the coding matrix. This then makes $l = 2^{k-1} - 1$. For example, if $k = 4.$, the coding matrix is given by

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{4.3}$$

It is easy to see that the minimum hamming distance is $\lceil (2^{k-1} - 1)/2 \rceil$. The disadvantage of exhaustive coding is that l increases exponentially with k.

4.2.3 Decoding Function

Decoding function determines how the distance between the outputs of base classifiers and codewords is calculated. A simple type of decoding function is to count the number of positions *s* in which the codeword entry differs from the sign of the prediction $f_s(\mathbf{x})$. Formally, the distance measure is given as

$$d_H(\mathbf{r},\mathbf{f}(\mathbf{x})) = \sum_{s=1}^{l}\left(\frac{1-sign(r_s f_s(\mathbf{x}))}{2}\right) \tag{4.4}$$

where $sign(z) = +1$ if $z > 0$, -1 if $z < 0$ and 0 if $z = 0$. r_s is the entry of codeword $\mathbf{r}$ at position *s*. This is called the *hamming distance* decoding. A disadvantage of this decoding function is that it totally ignores the output values of base classifiers. For binary classifiers like SVM, the magnitudes of outputs usually indicate the level of "confidence" of the prediction. For example, assume that the OVA coding matrix is used and some of the base classifiers are "weak"; then the weak classifiers may often give wrong signs of their predictions. Since the OVA codes do not have error correcting ability, a lot of errors may occur despite the fact that the other classifiers are "strong" and correct.

A second type of decoding function takes into account the confidence of predictions. It uses a loss function *L* which is algorithm-specific. The loss function calculates the "loss" of the prediction given the output values and the codewords. The loss function for SVM is defined as

$$L(y,f) = (1-yf)_+ \tag{4.5}$$

where *y* is an entry of the codeword and f is the output of the SVM. z_+ is defined as $\max(z,0)$. The distance measure can be written as

$$d_L(\mathbf{r},\mathbf{f}(\mathbf{x})) = \sum_{s=1}^{l}\left(L(r_s, f_s(\mathbf{x}))\right) \tag{4.6}$$

This is called the *loss based* decoding.

There is another type of decoding function that takes the prediction confidence into account by simply calculating the inner product of codewords and the vector of classifier outputs. The distance measure is defined as

$$d_I(\mathbf{r},\mathbf{f}(\mathbf{x})) = -\sum_{s=1}^{l}\left(r_s f_s(\mathbf{x})\right) \tag{4.7}$$

and it is called the *inner product* decoding.

Finally, we introduce a decoding function which is based on the probability of the prediction. Given the assumption that the base classifiers are independent, the probability that assigns sample $\mathbf{x}$ to class *r* is

$$J = \prod_{r_s=+1} p_s(\mathbf{x}) \prod_{r_s=-1} (1 - p_s(\mathbf{x})) \tag{4.8}$$

where p_s is the probabilistic output of the base classifier *s* and r_s ($s = 1, 2, \ldots, l$) are entries of the codeword for class *r*. The class of which the codeword gives the maximum joint probability is the one predicted. Negative log-likelihood can be used to define the decoding function as

$$d_P(\mathbf{r},\mathbf{p}(\mathbf{x})) = -\sum_{s=1}^{l}\frac{1+r_s}{2}\log(p_s(\mathbf{x})) - \sum_{s=1}^{l}\frac{1-r_s}{2}\log(1-p_s(\mathbf{x})) \tag{4.9}$$

where $\mathbf{r} = (r_1, r_2, \ldots, r_l)$ and $\mathbf{p}(\mathbf{x}) = (p_1(\mathbf{x}), p_2(\mathbf{x}), \ldots, p_l(\mathbf{x}))$. There is still a problem in using probabilistic decoding function. Classifier like SVM does not give probability of prediction directly. A parametric model can be used to estimate the probability as suggested by Platt (1999):

$$p_s(\mathbf{x}) = \frac{1}{1 + \exp(A_s f_s(\mathbf{x}) + B_s)} \tag{4.10}$$

where $f_s(\mathbf{x})$ is the output of the SVM which is trained as the base classifier s. The sigmoid parameters A_s and B_s can be found by maximizing the following log-likelihood function

$$H_s = \sum_i \log\left(\frac{1}{1+\exp(A_s f_s(\mathbf{x}_i)+B_s)}\right) \tag{4.11}$$

In which $\mathbf{x}_i$ are the samples that are involved in producing the outputs of the base classifier s. However, we cannot use the same samples to train the base classifier and to produce the outputs because the distribution of SVM outputs is very different from those training and testing samples. To address this problem, a three-fold cross-validation (CV) is used in our case to fit A_s and B_s. An additional advantage of probabilistic decoding is that it gives the probabilities of prediction.

4.2.4 Classification Methods

SVM is generally applied to binary problems. It often achieves superior performance than other machine learning techniques. Here, we use SVM as base classifiers in combination with output-coding scheme to solve multiclass cancer classification problems. Our SVM output-coding (SVM-OC) method is compared with three popular classifiers, the KNN, the multi-layer perceptron (MLP) neural network and the C4.5 decision tree. Those three methods have already been successfully applied to multiclass microarray classification by other researchers (Khan *et al*., 2001; Yeang *et al*., 2001; Tan and Gilbert, 2003).

A. Support vector machine

The main idea of binary SVM is to implicitly map data to a higher dimensional space via a kernel function and then solve an optimization problem to identify the maximum-margin hyperplane that separates training instances (Vapnik, 1998). The hyperplane is based on a set of boundary training instances, called support vectors. The optimization problem is formulated as an objective function which allows for non-separable data by penalizing misclassifications. New instances are finally classified according to the side of the hyperplane they fall into.

B. K-nearest neighbor

The main idea of KNN is that it treats all samples as points in n-dimensional space (n is the number of variables). Given a new testing sample $\mathbf{x}$, the algorithm classifies it by voting of K-nearest training samples as determined by some distance metric, typically Euclidian distance (Duda *et al.*, 2001).

C. C4.5 Decision tree

The decision tree algorithm is well known for its robustness and learning efficiency. The output of the algorithm is a decision tree, which can be easily represented as a set of symbolic rules. The learning algorithm applies a divide-and-conquer strategy to construct the tree. Each node of a decision tree is a test on the value of a gene and a leaf represents the class of a sample that satisfies the rest. The tree will return a "yes" or "no" decision when the sets of instances are tested. Rules can be derived from the tree by following a path from the root to a leaf and using the nodes along the path as preconditions for the rule, to predict the class at the leaf. The rules can be pruned to remove unnecessary preconditions and duplication (Duda *et al.*, 2001).

D. Backpropagation neural network

A MLP is a feed-forward NN with signals propagated only forwardly through layers of neurons. The MLP we used comprises of (i) an input-layer with gene expression data; (ii) a hidden-layer of neurons using tangent sigmoid transfer functions, and (iii) an output-layer of neurons using logistic sigmoid transfer functions. The tangent sigmoid and the logistic sigmoid transfer functions are defined, respectively, as

$$\text{tansig}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4.12}$$

and

$$\text{logsig}(x) = \frac{1}{1 + e^{-x}} \tag{4.13}$$

Each output neuron represents one cancer class. The prediction result corresponds to the class of the neuron with the largest output. All connections from neurons of one layer to neurons of the next layer have weights and biases. These weights and biases are initialized before training and can be adjusted by backpropagation training algorithm. Errors between network outputs and targets are calculated and then the gradient decent optimization algorithm is employed to backpropagate the errors so that the weights and biases can be adjusted to minimize the training errors (Duda *et al.*, 2001). All samples should be presented to the network once before weights and biases are modified. Such a procedure is called one epoch and it can be repeated until the performance goal is met.

E. Parameters for classification methods

The parameters for different classification methods are determined by three-fold CV to optimize performance and to avoid over-fitting. This is a simple but effective way of parameter selection. The only shortcoming of this method is the computational cost that incurs by multiple training and testing processes. For SVM, a linear kernel is used and the regularization parameter is set from $\{0.0001, 0.01, 1, 100, 10000\}$. For KNN, the number of neighbors K varies from one to the total number of training samples for a thorough optimization. For MLP NN, the number of neurons in the hidden-layer is chosen from $\{2, 5, 10, 20, 40\}$.

Backpropagation algorithm based on gradient descent with momentum and adaptive learning rate is employed. The number of epochs is set at 3000 with a mean-squared-error (MSE) performance goal of 10^{-6}, and the momentum constant is set at 0.9.

4.2.5 Feature Selection

There are three major categories of feature selection methods. The first one ranks the features according to their values and class labels. The highest-ranked features can be selected for follow-up training and testing. The second one is the dimension reduction method. It can reduce the number of features from thousands to dozens. Two well-established methods, the partial least squares (PLS) and the principal components analysis (PCA), are used. The PLS may incorporate, while the PCA may not incorporate, class labels into dimension reduction. The third category selects features by feedback from classifiers. Features that have least contribution to classification are eliminated so as to have a more compact feature subset and to enhance performance.

A. Gene ranking

Intuitively one would select those genes that are correlated with a class but are uncorrelated with the other classes. Li *et al.* (2004) have eight gene ranking methods for multiclass microarray classification. However, they did not find a clear winner out of the eight. We choose a gene ranking method which is based on the ratio of their between-group to within-group sums of squares. It is also used by a few other researchers as the gene selection method (Dudoit *et al.*, 2002; Lee and Lee, 2003). For a gene j, this ratio is defined as

$$BW(j) = \frac{\sum_i \sum_k I(y_i = k)(\bar{x}_{kj} - \bar{x}_{.j})^2}{\sum_i \sum_k I(y_i = k)(x_{ij} - \bar{x}_{kj})^2} \tag{4.14}$$

where $\bar{x}_{ij}$ and $\bar{x}_{kj}$ denote the average expression level of gene j across all classes and across samples belonging to class k only. $I(\cdot)$ is the indicator function. The base classifiers are built using genes with the largest BW values.

B. Dimension reduction

In microarray context, because the number of features is much larger than the number of samples, dimension reduction methods have been proposed to tackle the "curse of dimensionality" problem. It is prohibitive to use some of the statistical methods when $m < n$ because of excessive computational time. Dimension reduction is also used as the preprocessing step to make these methods feasible. The PLS (Wegelin, 2000) and the PCA (Golub and Van Loan, 1996) have been proven to be effective for microarray classification (Nguyen and Rocke, 2002; Shen and Tan, 2005a).

C. Recursive feature elimination

Given a sample $\mathbf{x}$, the linear SVM function can be written formally as

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{4.15}$$

where $\mathbf{w}$ is the weight vector and b is the bias. The elements of $|\mathbf{w}|$ thus indicate the contribution of the corresponding genes to the output of classification. The genes with the smallest values in $|\mathbf{w}|$ are dropped and this process can be executed recursively because each time a few genes are eliminated, the classifier should be run again to generate a new vector of $\mathbf{w}$. The method was first proposed by Guyon *et al*. (2002) to do feature selection in binary classification. In multiclass context, the RFE should be performed for each base classifier. As a comparison, Rifkin *et al*. (2003) also used SVM and RFE to carry out feature selection in multiclass microarray classification based on OVA coding strategy. However, we perform RFE in a slightly different way.

The RFE is executed on each base classifier independently so that the best performance and the smallest gene subset can be obtained concurrently for one base classifier. Three-fold CV is then used to evaluate the goodness of gene subset. Finally, it gives us a set of base classifiers from different subsets of genes respectively. This method is based on the assumption that all base classifiers can be considered as independent and we can thus optimize each classifier individually.

4.3 Results

4.3.1 Datasets and Experimental Setup

Two multiclass microarray datasets are selected for our experiments. The first is the GCM dataset published by Ramaswamy *et al*. (2001). It consists of 144 training samples and 54 testing samples of 15 common cancer classes. Each sample has 16063 gene expression levels. The data is available at: http://www-genome.wi.mit.edu/MPR/GCM. For simplicity, we dropped the eight metastatic samples from the testing dataset because they are not present in the training dataset. Therefore, 46 testing samples and 14 cancer classes are considered. The distribution of training and testing samples among the 14 classes is listed in Table 4.1. The second is the ALL dataset published by Yeoh *et al*. (2002). It consists of 163 training samples and 85 testing samples of 6 subtypes of acute lymphoblastic leukemia. Each sample has 12558 gene expression levels. The data is available at: http://www.stjuderesearch.org/data/ALL1/. The distribution of training and testing samples among the six classes is listed in Table 4.2. All data is log-transformed and all genes are normalized to have zero mean and unit standard deviation. No other preprocessing steps are applied.

For GCM data, three coding strategies are used: AP, OVA, and random. We did not use exhaustive coding because l would be equal to $2^{13}-1=8191$ and this will

Table 4.1 GCM: number of samples per cancer class.

Cancer Class	Training	Testing
Breast (BR)	8	3
Prostate (PR)	8	2
Lung (LU)	8	3
Colorectal (CO)	8	3
Lymphoma (LY)	16	6
Bladder (BL)	8	3
Melanoma (ME)	8	2
Uterus (UT)	8	2
Leukemia (LE)	24	6
Renal (RE)	8	3
Pancreas (PA)	8	3
Ovary (OV)	8	3
Mesothelioma (ML)	8	3
Brain (CNS)	16	4

Table 4.2 ALL: number of samples per subtype.

Subtype	Training	Testing
BCR-ABL	9	6
E2A-PBX1	18	9
Hyperdiploid>50	42	22
MLL	14	6
T-ALL	28	15
TEL-AML1	52	27

make the computation intractable. For ALL data, AP, OVA and exhaustive coding strategies are used.

According to the suggestion of Li. *et al*. (2004), 250 top genes are selected from BW ratio ranking. We also tested the data without feature selection, which is denoted as NO in Tables 4.5 and 4.6. For RFE, the gene subset for each base classifier is determined by three-fold CV. For PLS, components are extracted so that 99% of the variances of response variables, and 80% of the variances of predictor variables are captured. For PCA, components are sorted in descending order according to their eigenvalues and the first ones are selected so that 80% of the variances of predictor variables are captured. The percentage is determined empirically and is considered sufficient for classification, as a rule of thumb.

All programs are written in MATLAB® codes. The software package written by Steve Gunn is used for the SVM algorithm. It is available at: http://www.kernel-machines.org/. However, we did some modifications so that the speed is enhanced.

4.3.2 Testing on Output-Coding and SVM

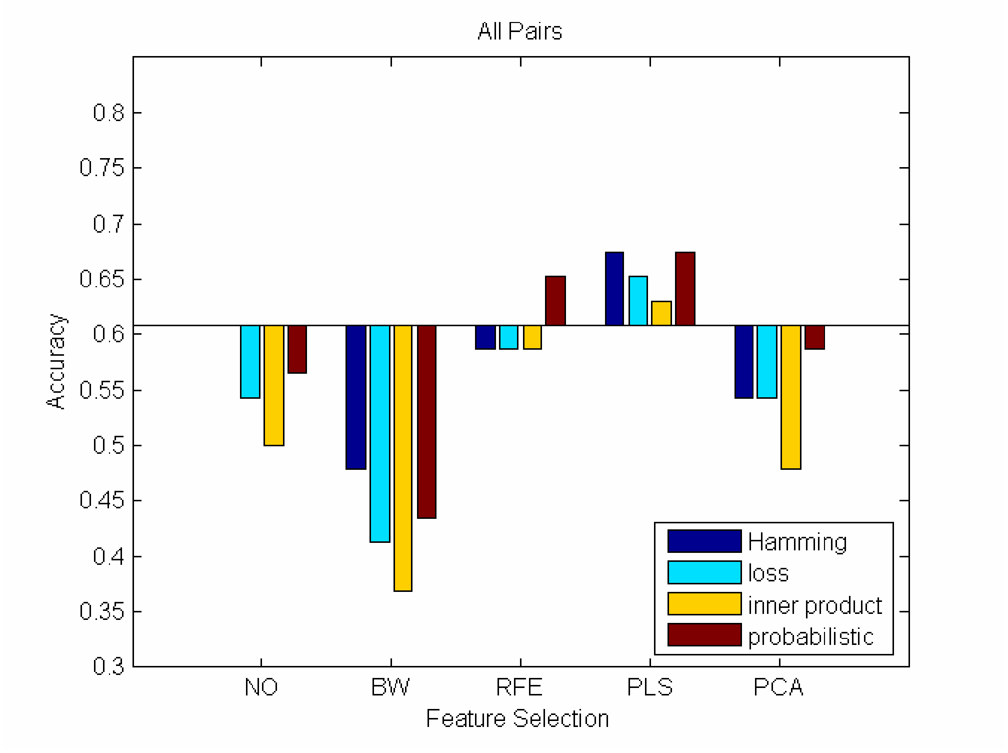
All combinations of coding strategies, decoding functions and feature selection methods with SVM are applied on the ALL and GCM datasets. The results are shown in Figures 4.1 and 4.2. In (a)-(c) of Figures 4.1 and 4.2, the base line value is equal to that of the output-coding without feature selection and. For AP and ECOC codings, decoding is hamming-distance; for OVA coding, decoding is inner-product. In (d) of Figures 4.1 and 4.2, the maximum and mean accuracies of all coding strategies are given; the probabilistic decoding fails to produce proper outputs with certain codings and feature selections, and is thus not included. The corresponding accuracies are also truncated in (a)-(c) of Figures 4.1 and 4.2. More information is given in Table 4.3. We have the following observations:

1. The ECOC coding strategy generally outperforms the other coding strategies. The highest accuracy on the GCM data is achieved by random coding. Combining with loss based and inner product decoding functions and RFE, a 80.4% testing accuracy has been obtained. On the ALL data, exhaustive coding has achieved almost perfect accuracy for most decoding functions and feature selection methods. There are some exceptions on probabilistic decoding function. This can be attributed to the ability of ECOC to correct errors for weak base classifiers.

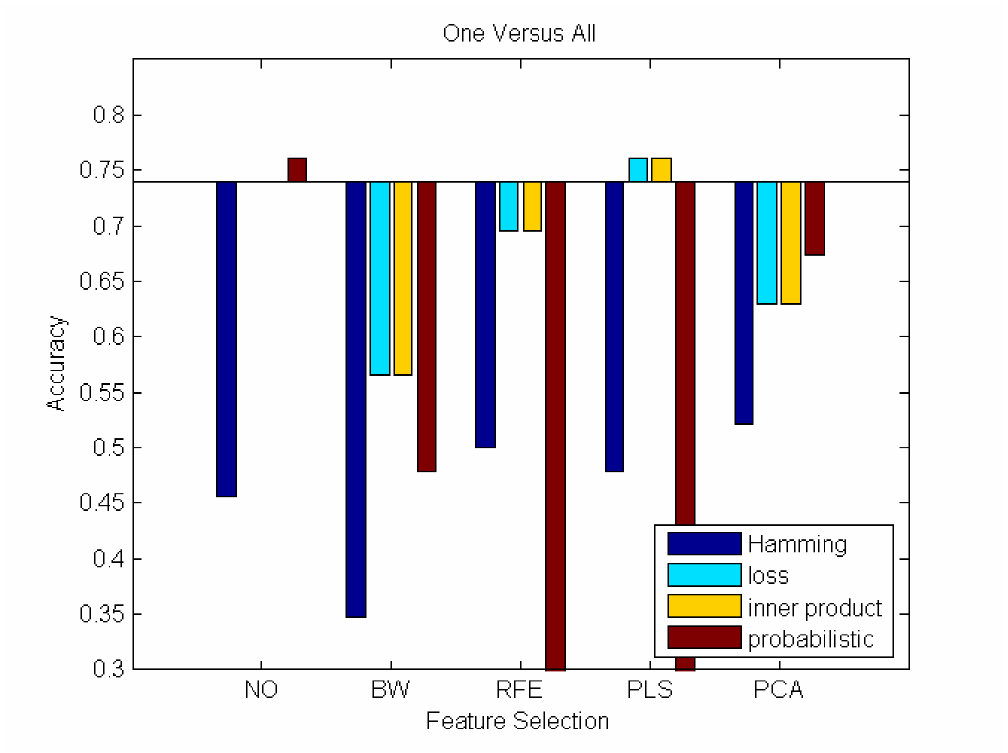
2. The AP coding strategy works rather well with the ALL data, but it is the worst coding strategy with the GCM data. All testing accuracies on GCM data are below 70%. Learning from Table 4.1, we know that some classes of the GCM data are very small. It is hard for base classifiers to perform well on those pairs of small cancer classes. A lot of errors may occur for base classifiers; thus the multiclass classification accuracy is degenerated.
3. The probabilistic decoding function is very sensitive to coding strategies and feature selections. It can be observed from Table 4.3 that it fails to work with RFE and PLS when OVA coding strategy is applied to GCM data. It also fails when OVA and exhaustive coding strategies, and BW, RFE and PLS feature selections are used on the ALL data. However, it achieves 100% accuracy on the ALL data when AP and BW are used. It is known that fitting sigmoid parameters by solving (4.11) is sensitive to the distribution of samples of two classes. If the negative class is very large but the positive class is very small, (4.11) would simply produce an infinite negative A_s, and vice versa. For microarray data, the unequal distribution of classes may cause the failure of probabilistic decoding function. It is observed that the probabilistic decoding function never fails with the AP and random coding strategies when the distributions of these two classes trained by base classifiers are more balanced than those of the OVA and exhaustive coding strategies.
4. The hamming-distance decoding function is not suitable for OVA. This is because many ties will happen when the base classifiers do not give enough high prediction confidence and they are just solved by random assignment. It is better to integrate prediction confidence when OVA is used. However,

hamming-distance decoding works well with AP and ECOC. This is because the base classifiers of AP usually have high prediction confidence and ECOC has the ability to correct errors if base classifiers are weak. It is noticed that loss-based and inner product decoding give very similar results.

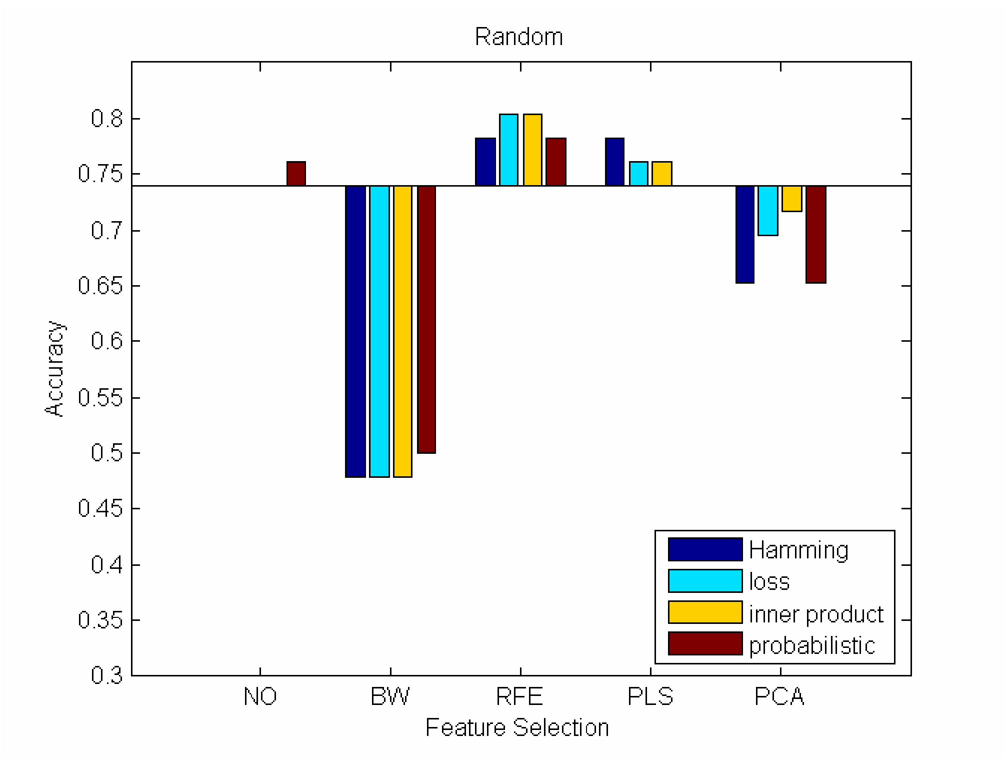
5. Feature selection by BW ratios performs poorly with GCM data but rather well with ALL data. This is consistent with the results by Li *et al*. (2004). BW only gives the values between and within group variances but no information about the class labels. It may select genes that only contain information of several classes without regards to the rest. Again, this is related to the class-unbalance condition of GCM. A possible alternative is to use the student T-statistics to select genes that distinguish one class from the rest and make a uniform distribution for the numbers of genes for all classes. It is also noticed that results are usually good when no feature selection is used.
6. PLS outperforms PCA no matter what coding matrix and decoding function are used, excluding only a few exceptions from probabilistic decoding function. It has been validated that PLS is usually a better dimension reduction method than PCA for supervised classification (Nguyen and Rocke, 2002; Shen and Tan, 2005a), mainly because PLS incorporates class labels in generating its components. 32 PLS components are extracted for both GCM and ALL datasets while 37 and 80 PCA components are extracted for the above two datasets, respectively. With fewer components, the training and testing speeds can usually be enhanced.


All Pairs
Accuracy
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
NO
BW
RFE
PLS
PCA
Feature Selection
Hamming
loss
inner product
probabilistic

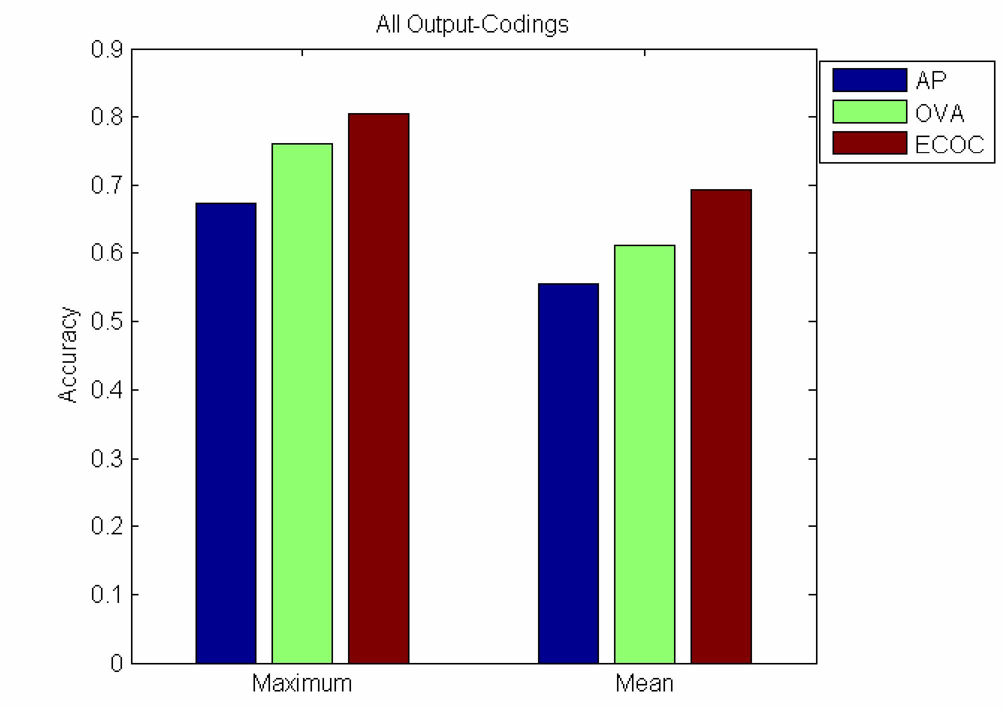
(a)


One Versus All
Accuracy
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
NO
BW
RFE
PLS
PCA
Feature Selection
Hamming
loss
inner product
probabilistic

(b)

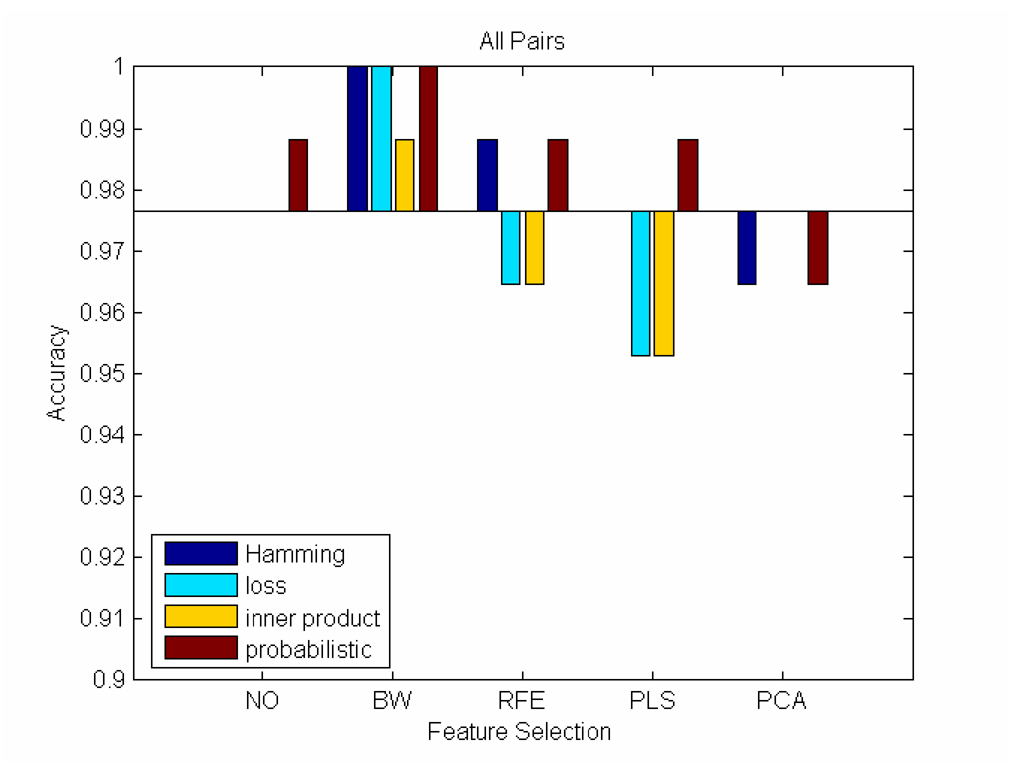


(c)

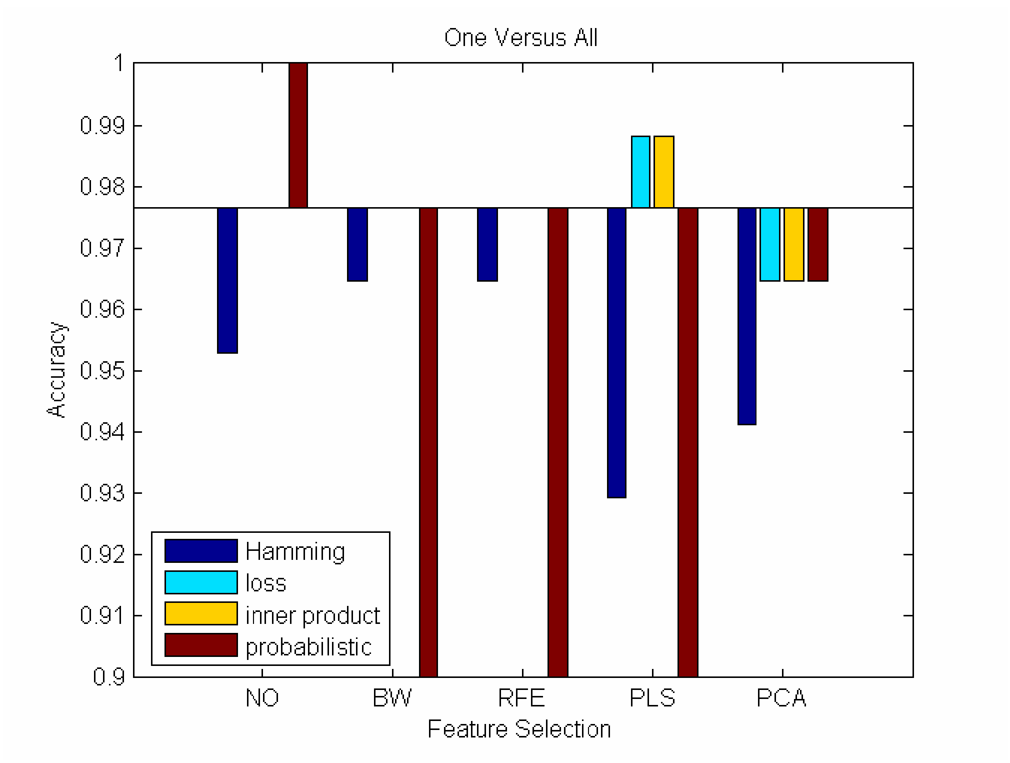


(d)

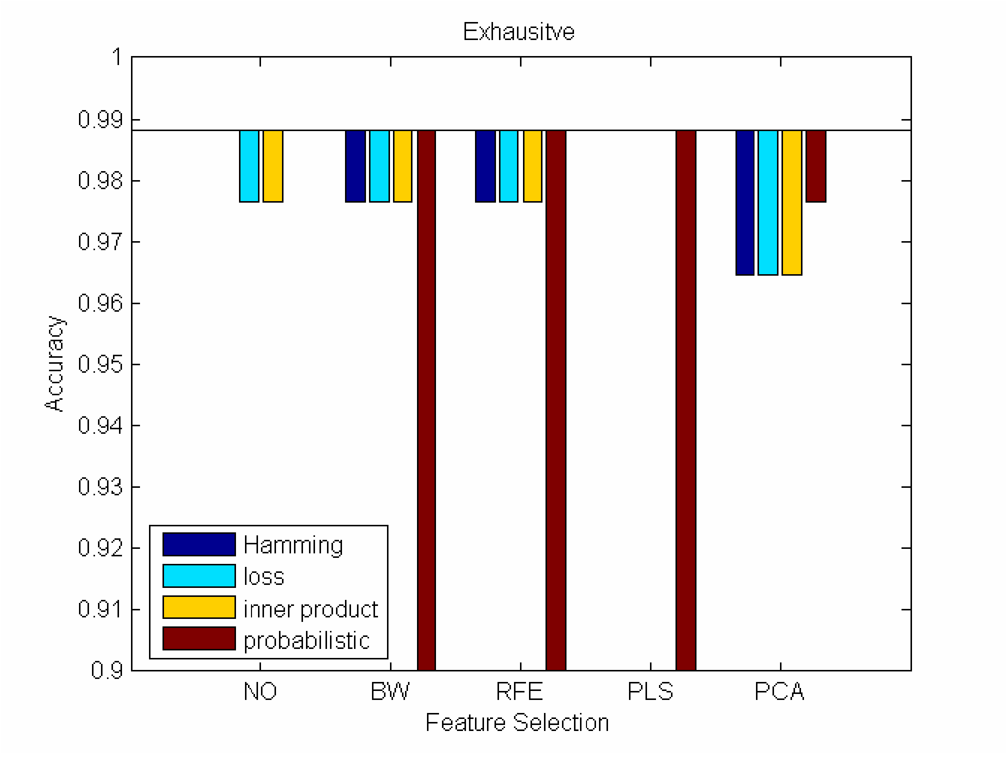
Figure 4.1 Accuracies of output-coding on GCM data using different coding strategies, decoding functions and feature selections.


All Pairs
Accuracy
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.9
Hamming
loss
inner product
probabilistic
NO
BW
RFE
PLS
PCA
Feature Selection

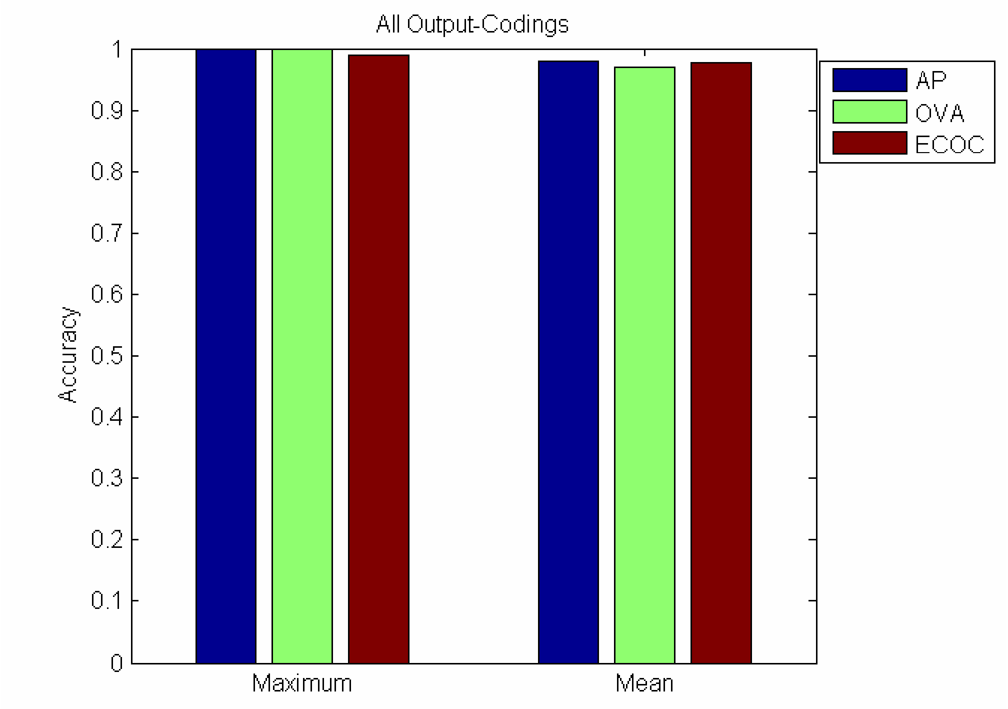
(a)


One Versus All
Accuracy
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.9
Hamming
loss
inner product
probabilistic
NO
BW
RFE
PLS
PCA
Feature Selection

(b)



(c)



(d)

Figure 4.2 Accuracies of output-coding on ALL data using different coding strategies, decoding functions and feature selections.

Table 4.3 The combinations that the probabilistic decoding fails to produce proper outputs.

Dataset	Coding strategy	Feature selection	Errors
GCM	OVA	RFE and PLS	43
ALL	OVA	BW, RFE and PLS	79
ALL	Exhaustive	BW, RFE and PLS	79

Another experiment is performed to find out the relationship between the accuracy of output coding scheme and the codeword length l on GCM data. Random coding strategy and RFE are used. We vary l at $\lceil 5\log_2 k \rceil$ and from $\lceil 10\log_2 k \rceil$ to $\lceil 40\log_2 k \rceil$. The results are plotted in Figure 4.3. The highest accuracy is 83% when $l = \lceil 20\log_2 k \rceil$, using loss based and probabilistic decoding functions. Increasing the codeword length does not necessarily increase the accuracy. This is because some base classifiers corresponding to the codeword bits are weak and may commonly introduce errors. They simply degrade the performance without increasing the error correcting power of ECOC. This is confirmed by deleting the codeword bits that correspond to the base classifiers with CV errors above average. It is done on codeword lengths $\lceil 30\log_2 k \rceil$ and $\lceil 40\log_2 k \rceil$. Comparison of results before and after deletion is listed in Table 4.4.

4.3.3 Comparison of Classification Accuracies with Other Classification Methods

To compare the classification accuracies of SVM-OC with those of the other classification methods, three popular classifiers, KNN, C4.5 and MLP were applied to the GCM and ALL datasets. Three-fold CVs were used on the training data to find the optimal parameters and then the trained classifiers were applied to the testing data. Three feature selection methods, BW, PLS and PCA were used. RFE was not used

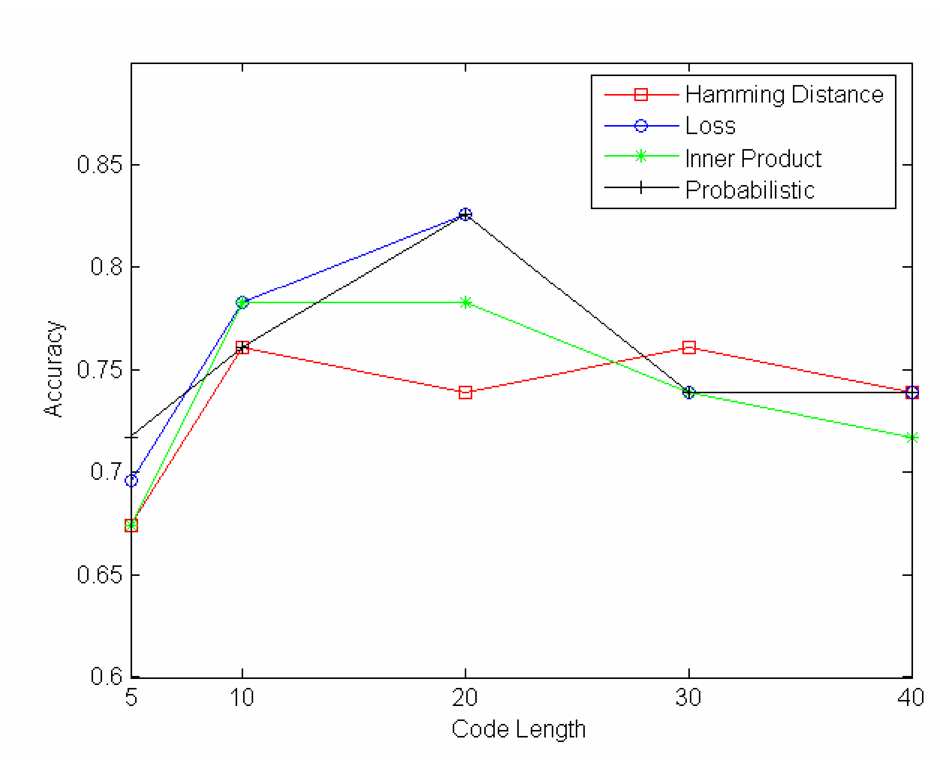


Figure 4.3 Accuracy versus random code length.

Table 4.4 Testing errors before and after deletion of codeword bits.

Codeword Length	$l = \lceil 30\log_2(k) \rceil$		$l = \lceil 40\log_2(k) \rceil$	
Decoding Function	Before	After	Before	After
Hamming Distance	11	10	12	11
Loss Based	12	10	12	10
Inner Product	12	10	13	12
Probabilistic	12	10	12	11

because it is not possible to combine it with KNN and C4.5. For SVM-OC, we cited the results from Section 3.2 and they are given in Tables 4.5 and 4.6.

It can be seen that the SVM-OC achieves the best performance no matter which feature selection method is used. For the other three classifiers, MLP seems to be better. It outperforms KNN and C4.5 in most cases, and achieves very similar results with SVM-OC on ALL data. The classification accuracies of SVM-OC cannot be enhanced further by feature selection methods. However, the performance of KNN and MLP can usually be significantly improved if a proper feature selection is

Table 4.5 Testing accuracies in percent on GCM dataset.

Classification Method	NO	BW	PLS	PCA
SVM-OC	**76.1**	**56.5**	**78.3**	**71.7**
KNN	47.8	37	63	37
C4.5	52.2	39.1	47.8	43.5
MLP	65.2	47.8	73.9	60.9

Table 4.6 Testing accuracies in percent on ALL dataset.

Classification Method	NO	BW	PLS	PCA
SVM-OC	**100**	**100**	**98.8**	**97.6**
KNN	87.1	100	89.4	88.2
C4.5	81.2	81.2	85.9	84.7
MLP	95.3	97.6	97.6	95.3

employed. For example, accuracy is improved by 8.2% on GCM when PLS is used for MLP; accuracy is also improved by 12.9% on ALL when BW is used for KNN. KNN has large variance of the prediction in high-dimensional spaces because all the training points are located close to the edge of the sample (Hastie *et al*., 2001). Furthermore, many irrelevant variables dominate the distances between samples which cause serious problems for the prediction (Mitchell, 1997). MLP suffers badly from the "curse of dimensionality" because of at least two reasons: (i) There may be more local minima in the error spaces so that backpropagation may easily fall into one of them. (ii) The model space increases exponentially with the number of variables; therefore, it becomes difficult to find a model that generalizes. C4.5 is the only nonmetric method among the four. This unique decision tree algorithm generates a set of symbolic rules from which the predictions on new testing samples are made. Generalization can often be improved by pruning the minimum impurity leaves; thus the method is insensitive to high dimensionality.

4.4 Discussions and Summary

The output coding scheme from machine learning has been successfully applied to multiclass microarray classification. Usage of different coding matrices, decoding functions and feature selection methods have been discussed. Combining ECOC, RFE and some decoding functions, 83% testing accuracy has been achieved on the GCM data. Comparing with other machine learning methods, SVM-OC has shown its superior performance and is much less sensitive to the “curse of dimensionality”.

It has been shown that a good coding matrix can result in high accuracy of multiclass microarray classification. Better coding strategies are required to further improve the performance of the output coding scheme. Crammer and Singer (2001) have reported that for a given fixed l, finding a $k \times l$ coding matrix $\mathbf{M}$ which minimizes the empirical loss is NP-complete. We have empirically shown that by deleting the codeword bits corresponding to base classifiers of high CV errors, the multiclass accuracy can usually be improved. This stimulates the idea that code-matrix can be recursively improved as follows: (i) Start with a coding matrix of sufficiently large l. (ii) Delete the codeword bits corresponding to high CV errors. (iii) Use the hill-climbing algorithm to improve the row separation of truncated coding matrix. This procedure can be iterated for a number of times until good results have been obtained. The computational cost of this heuristic method, however, would be high.

Although gene ranking and dimension reduction methods have been shown to be effective for multiclass classification, it is observed that sometimes without feature selection, the results are even better. RFE is good for binary classification but for output coding based multiclass classification, it can only be used to enhance base classifiers. Data over-fitting can easily happen and the variances of outputs would be large especially when class sizes are small. This can degrade the multiclass accuracy in the end. It is better to use the CV errors of multiclass classification as feedback to select genes. Some algorithms like genetic algorithm could be considered.

Chapter 5. Optimized Output-Codes from Genetic Algorithm for Multiclass Cancer Classification

5.1 Introduction

In the spirit that the multiclass problems can be solved in a divide and then combine way, the error-correcting output-coding (ECOC) has been proposed to enhance the accuracy of ensemble classifier (Dietterich and Bakiri, 1995). In their work, each class is assigned with a unique codeword. As long as two codewords have sufficiently a large hamming-distance, the ensemble classifier is able to correct some errors made by binary classifiers at the combination stage. ECOC has been successfully applied to cancer classification and is shown to be reliable and accurate (Valentini, 2002; Li *et al.*, 2004; Shen and Tan, 2005b).

Each codeword represents a row of the coding matrix. The primary objective in designing a coding matrix is to jointly maximize its row separation as well as its column separation. Since the entries of the coding matrix are discrete, the problems is known to be NP-hard (Crammer and Singer, 2001). Therefore, it is not practical to test all possible random coding matrices to find the best one. In addition, the codeword length is variable. We may need to try different codeword lengths for a specific problem. Genetic algorithm (GA) (Goldberg, 1989) provides a simple but effective approach to solve optimization problems that cannot be solved in a deterministic way as in standard optimization problems. It is thus very natural to use GA to find the coding matrix that is optimal for a given criterion.

In this work, a new criterion has been proposed to evaluate the fitness of a given coding matrix. GA is then used to find the coding matrix with the optimal fitness.

5.2 Methods

5.2.1 Genetic Algorithms to Generate Output-Codes

Experiments have shown that both error recovering power and independence among codeword bits contribute to the accuracies of the output coding method (Masulli and Valentini, 2003). It is of particular interest to jointly optimize the row separation and the column separation of a coding matrix. A discouraging fact is that the number of possible coding matrices is exponential with the matrix dimensions. Therefore, it is unrealistic to enumerate all random matrices and choose the one that satisfies our criterion.

GA has shown to be a powerful method to search in a space in order to find the global minimum (Grefenstette and Baker, 1989). Therefore, it may be advantageous to use GA over other approaches to achieve our goal. The pioneer work has been reported by Kuncheva (2005), in which it was argued to use diversity measure (DIV) instead of minimum hamming-distance (HAM) as a criterion for output-codes generation. However, real datasets were not used in his work. Instead, only simulated binary classifiers were used, which were assumed to have the same error rates and randomly made errors according to a uniform distribution. Based on this assumption, it was claimed that DIV was superior to HAM.

First, let us introduce Kuncheva's work. Given a coding matrix $\mathbf{M}$ ($k \times l$), its entries only have values 1 and –1. The row separation is defined as the Hamming-distance between two rows i, j,

$$R_{ij} = \sum_{s=1}^{l} \frac{|\mathbf{M}(i,s) - \mathbf{M}(j,s)|}{2} \tag{5.1}$$

where $\mathbf{M}(i,s)$ and $\mathbf{M}(j,s)$ are the sth elements of row i and j, respectively. The column separation is defined slightly different. If column j is complement to column

i, the codes of the two columns are essentially the same because they infer the same separation line between classes. The column separation is thus defined as

$$C_{ij} = \min\left\{\sum_{s=1}^{k}\frac{|\mathbf{M}(s,i)-\mathbf{M}(s,j)|}{2}, \sum_{s=1}^{k}\frac{|\mathbf{M}(s,i)+\mathbf{M}(s,j)|}{2}\right\} \tag{5.2}$$

Therefore, the normalized minimum hamming-distance among all rows of a coding matrix is given by

$$H_r = \min_{1\le i,j\le k}\left\{R_{ij}/l\right\} \tag{5.3}$$

and among all columns is given by

$$H_c = \min_{1\le i,j\le l}\left\{C_{ij}/k\right\} \tag{5.4}$$

The normalized averaged hamming-distance for all rows of a coding matrix is

$$D_r = \frac{2}{k(k-1)}\sum_{i<j} R_{ij}/l \tag{5.5}$$

and for all columns is

$$D_c = \frac{2}{l(l-1)}\sum_{i<j} C_{ij}/k \tag{5.6}$$

Finally, the HAM and the DIV are defined as

$$H = \frac{H_r + H_c}{2} \tag{5.7}$$

and

$$D = \frac{D_r + D_c}{2} \tag{5.8}$$

respectively. Please note that the hamming-distances are normalized by vector lengths so that H_r, H_c, D_r and D_c are defined to be length independent.

The objective is to find a coding matrix which has the maximum H or D value. GA solves an optimization problem by imitating natural selection (Goldberg, 1989). Each coding matrix is treated as a “chromosome”, with its rows concatenated into a bitstring of length kl. The procedure starts with a randomly generated population of a fixed size. The chromosomes are evaluated by fitness functions. For this case, $fitness = -H$ or $-D$. The smaller the fitness values, the more probable the chromosomes are selected as parents. Children are produced by making random changes to a single parent — which is guided by "mutation" rule, or by combing a pair of parents — which is guided by "crossover" rule. After a number of children have been produced, the current population is replaced with the children to form the next generation. The size of the children should equal the size of the parents. This process can be iterated for sufficient times until both of the mean and the minimum fitness values of the population are minimized. Finally, the chromosome with the minimum fitness value in the last population is selected. This chromosome is then converted back to a coding matrix which should have the maximum H or D value.

A phenomenon has been noticed, for which cautions should be taken. For example, the coding matrix, $\mathbf{M}_f$, in (5.9) has $D = 0.33$:

$$\mathbf{M}_f = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \end{bmatrix} \tag{5.9}$$

However, since the second column is complement to the first column and the third and the fourth columns contain 1 or –1 only, all of them should be removed from the original coding matrix, resulting in a reduced coding matrix with only one column. This phenomenon is named as "false fitting". This also occurs with HAM. It can be solved by calling a procedure which removes such kind of columns before H and D are calculated. This will produce a coding matrix with a reduced size. However, the codeword lengths l in (5.3) and (5.5) remain unchanged. Therefore, the "false fitting" coding matrices will be penalized, producing larger fitness values and reduced chances to be selected as parents by GA.

5.2.2 Multiple Margins for Coding Matrix

The ultimate goal in designing a coding matrix is to enhance the ensemble classifier performance. A training error upper bound of the output-coding has been proved.

Theorem 1 (Allwein *et al.*, 2000): Let ε be the average binary loss of hypotheses $f_1,\ldots,f_l$ on a given training set $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_m,y_m)$ with respect to the coding matrix $\mathbf{M}\in\{-1,0,+1\}^{k\times l}$, where k is the number of classes. Let ρ be the minimum hamming-distance of all rows as defined in (5.3). Then the training error, e, for output-coding has the upper-bound

$$e \le \frac{l\varepsilon}{\rho C} \tag{5.10}$$

where C is a constant that only depends on the decoding function.

This gives rise to the HAM which maximizes ρ. However, it is too loose to consider the worse case only in practice. This is why DIV is advocated by Kuncheva (2005). The coding matrix produced by DIV measure may make more errors on the samples in the two closest classes, but it may perform better on average. This may sound good for DIV generated coding matrices, but it should also be tried to avoid constantly making errors on certain classes. A tighter error bound is derived, which may lead to better coding matrices.

At first, let us define a "margin" for class i

$$\eta_i = \min\left\{R_{ij},\ 1 \le j \le k,\ j \ne i\right\},\ 1 \le i \le k \tag{5.11}$$

We then have the following theorem.

Theorem 2: Let all symbols be defined as above and assume the sizes of all classes are $m_1, \ldots, m_k$, where $m = m_1 + \cdots + m_k$. The training error, e_m, has an upper-bound

$$e_m \le \frac{l\varepsilon}{mC}\left(\frac{m_1}{\eta_1} + \cdots + \frac{m_k}{\eta_k}\right) \tag{5.12}$$

and we have

$$e_m \le e \tag{5.13}$$

Proof: Following the results in **Theorem 1**, it can easily be seen that the training error of class i, e_i, has the upper-bound

$$e_i \le \frac{l\varepsilon}{\eta_i C} \tag{5.14}$$

since all training errors are induced by the k classes, the overall training error is the weighted average of each individual class:

$$e_m = \frac{1}{m}\left(m_1 e_1 + \cdots + m_k e_k\right) \tag{5.15}$$

Incorporating (5.14) into (5.15), (5.12) is thus proved. Since $\rho \le \eta_i,\ 1 \le i \le k$, we have

$$\frac{l\varepsilon}{\eta_i C} \le \frac{l\varepsilon}{\rho C}, \qquad 1 \le i \le k$$

This implies $\quad \dfrac{l\varepsilon}{mC}\left(\dfrac{m_1}{\eta_1} + \cdots + \dfrac{m_k}{\eta_k}\right) \le \dfrac{l\varepsilon}{m\rho C}\left(m_1 + \cdots + m_k\right)$

Since $\quad m = m_1 + \cdots + m_k$

Then $\quad \dfrac{l\varepsilon}{mC}\left(\dfrac{m_1}{\eta_1} + \cdots + \dfrac{m_k}{\eta_k}\right) \le \dfrac{l\varepsilon}{\rho C}$

(5.13) is thus proved. □

In maximizing all margins simultaneously, the mean of all margins is maximized. As the counterpart of H_r and D_r, M_r is defined as

$$M_r = \frac{1}{2k}\sum_{i=1}^{k}\frac{\eta_i}{l} \tag{5.16}$$

Besides row separation, the column separation should also be maximized to reduce the average binary loss. Therefore, the new fitness function, "multiple margins" (MLM), for coding matrix generation is defined as

$$M = \frac{M_r + D_c}{2} \tag{5.17}$$

where D_c is defined as in (5.6) and the name, MLM, is to reflect the fact that there are several margins for all classes.

5.3 Results

5.3.1 Datasets and Parameters Setup

Two commonly used cancer datasets have been selected to test output-coding methods. The GCM data contains 14 primary tumor types, 218 samples and 15009 genes. The NCI60 data contains 9 tumor types, 60 samples and 5726 genes. They were both produced by oligonucleotide microarrays. The datasets are publicly available and can be downloaded from the website described in a recent classifiers benchmarking paper (Statnikov *et al.*, 2005). The Golub's signal-to-noise method (Golub *et al.*, 1999) is used for gene selection in an OVA fashion. The selected gene expression profiles are then normalized to have zero mean and unit standard deviation.

Linear SVM is used as a binary classifier. The regularization parameter is fixed to one for all the experiments. There are numerous parameters that can impact the performance of GA. The best set of parameters is rather problem-specific. Let $k = 9$ and $l = 9$ and the GA is tested with different parameters. There are the following observations. Both selection methods, “stochastic uniform” and “roulette”, give good results. For the crossover function, “scattered” and “single point” yield good results, while “heuristic” gives the worse results. The mutation rate should be set as a small value. The crossover fraction should be set to less than 1.0. However, a high fraction like 0.8 would give good results. The parameter settings used in this work are summarized in Table 5.1. Figure 5.1 shows an illustration of the evolution of fitness values using MLM. It can be seen that the fitness value drops drastically at the first several generations. It then reaches a stable point; this implies a local minimum has been found.

One important concern of GA is the reproducibility and reliability. To tackle this problem, GA is repeated for 100 times with different initial states to generate the coding matrices using MLM, DIV and HAM criteria. The means and standard deviations of the inverse final fitness values and convergence time and the medians of generations are given in Table 5.2. It can be seen that the fitness values have very small variances, indicating that GA converges to the same local minimum often. The time used by GA is not subject to large fluctuations but is within a reasonable range. GA stops evolving if it cannot improve the fitness values for a given period of time or a given number of consecutive generations. The experiment shows that GA is not sensitive to the initial random states in both aspects of fitness and time.

Table 5.1 Parameter settings for GA.

Parameter	**Value**
Population Size	100
Selection Function	Roulette
Crossover Function	Single Point
Crossover Fraction	0.8
Mutation Function	Uniform
Mutation Probability	0.01
Generations	100

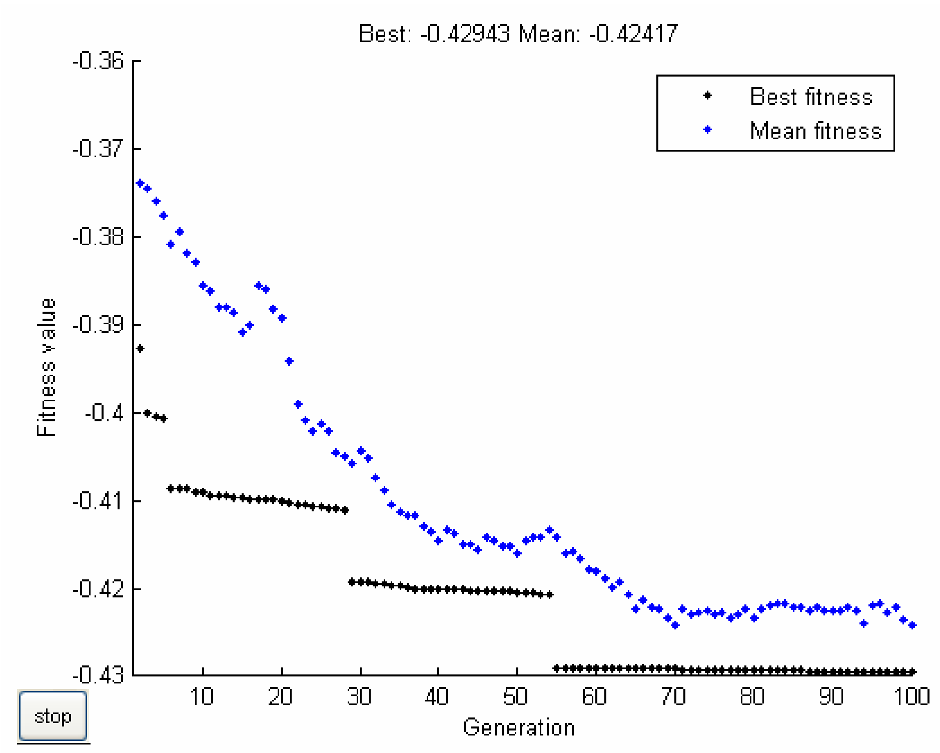


Figure 5.1 Using GA to find a coding matrix with maximum MLM.

Table 5.2 GA is repeated for 100 times to generate coding matrices: means and standard deviations of fitness and time. Median of generations.

Criterion	Inverse fitness	Time (s)	Generations
MLM	0.8810±0.0133	36.88±6.39	80.5
DIV	0.9783±0.0042	39.87±4.13	97.5
HAM	0.6433±0.0657	28.20±6.21	58.5

5.3.2 Pearson Correlations of Matrix Measures with Classification Errors

To investigate whether the above criteria indeed relate to the output-coding performance, an experiment was designed to find out the Pearson correlations of the fitness values, $-M$, $-D$, $-H$, with the classification error rates. One hundred 9×9 pure random matrices were generated as the coding-matrices for classification of NCI60 cancer dataset. Three-fold cross-validation (CV) was used to assess the accuracies of coding matrices. To reduce bias and variance, it was repeated for 100 times and the accuracies were averaged. Fifty genes were selected for each class. The Pearson correlation ρ and their P-values were calculated. The results are summarized in Table 5.3. It can be seen that the MLM criterion has the largest ρ with a very small P-value. It indicates a strong opposition against the null hypothesis (assume no correlation). The DIV has slightly smaller ρ than the HAM. This seems to contradict with Kuncheva's hypothesis (Kuncheva, 2005). It shows that testing the coding-matrices using synthetic binary classifiers may not be conclusive in real applications. Figure 5.2 gives a cluster plot of M values vs. mean errors.

Table 5.3 Pearson correlations of coding matrix criteria with classification error.

Fitness	ρ	P-value
$-M$	0.562	1.214e-009
$-D$	0.456	1.861e-006
$-H$	0.492	1.970e-007

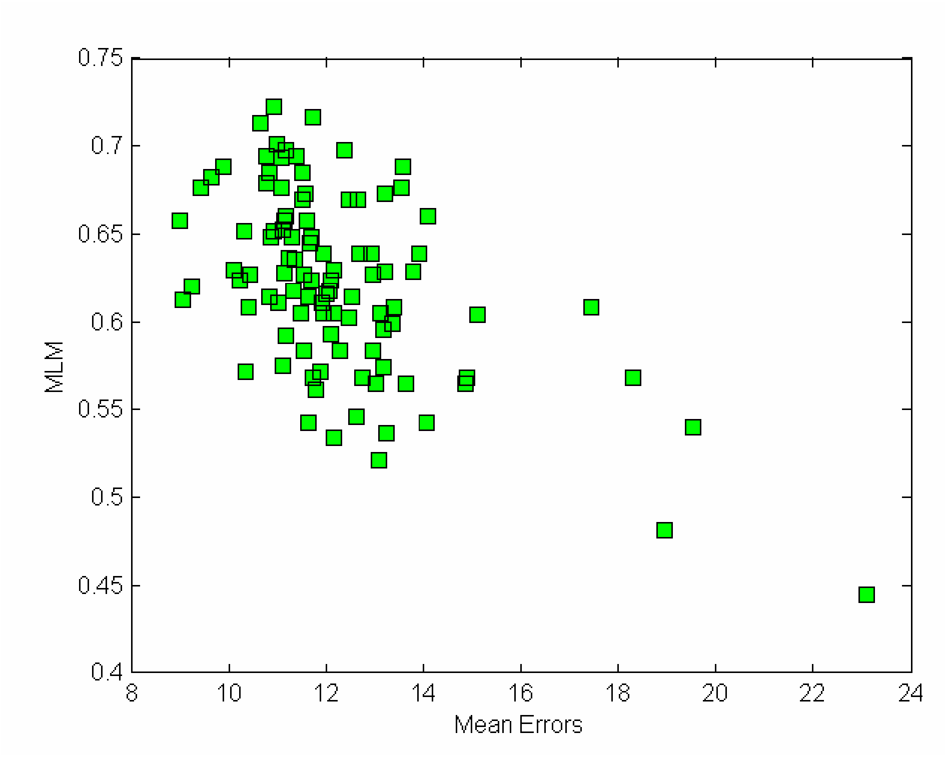


Figure 5.2 Cluster plot of MLM values vs. mean classification error rates.

5.3.3 Classifiers Benchmarking

To explore how the output-coding methods compared with the other methods, the experiments are designed as follows. The sizes of gene subset are set to 20, 50 and 100 for each class. Three-fold CVs are performed and repeated for 100 times. To infer the statistical significance of classifier difference, both the means and the standard deviations are recorded. The two classic coding methods, OVA and AP are chosen. In addition, two other non-coding methods, K-nearest neighbours (KNN) and C4.5 decision trees (C45) are used for comparisons. For output codes, let $l = \lfloor 10 \times \log_2(9) \rfloor = 31$ for NCI60 or $l = \lfloor 10 \times \log_2(14) \rfloor = 38$ for GCM. As a comparison, a randomized hill-climbing (RHC) method (Ricci and Aha, 1998) is also employed to enhance the purely randomly generated coding matrices. The unnormalized matrix measures of the GA generated and RHC enhanced coding matrices are given in Table 5.4. It can be learned from Table 5.4 that the matrix measures are actually correlated. Though MLM is used to find matrices with the maximum M_r, it also manages to find the matrices with the maximum H_r. DIV finds the matrices with the maximum D_r, as it is intended. HAM and RHC are good at finding the maximum H_c. However, these results are not surprising. A matrix with a large H_r often implies a large D_r and a large M_r, and vice versa.

Table 5.4 Matrix measures (unnormalized) of the coding matrices.

Matrix	NCI60				GCM			
measure	MLM	DIV	HAM	RHC	MLM	DIV	HAM	RHC
H_r	16	15	13	11	18	17	16	14
D_r	16.78	17.22	16	15.22	19.89	20.26	19.31	19.34
M_r	16.11	15.67	13.33	13	18.29	17.71	16.21	15.21
H_c	1	1	1	2	1	1	3	2
D_c	3.39	3.38	3.33	3.37	5.74	5.79	5.67	5.58

The testing errors are summarized in Tables 5.5 and 5.6. In the following, OVA, AP, INNR, HAMM, LOSS and PROB are defined as in Chapter 4. For OVA, the class label is assigned to the corresponding binary classifier that has the largest output. This equals INNR decoding function. For AP, the class label is based on the maximum binary votes, which equals HAMM decoding function. The PROB decoding is also used for OVA and AP. Generally, the output-coding classifiers using GA generated codewords outperform all the other methods. Among them, MLM achieves the best performance while DIV performs slightly better than HAM and RHC. This can be explained by the Pearson correlations between matrix measures and ensemble classifier errors. It is noticed that OVA gives very competitive results. The coding matrices of OVA do not have high values if measured by HAM, DIV or MLM. However, the binary classifiers for OVA are much stronger than the other coding methods. As another comparison, AP constructs separation planes between each pair of classes. It should also be able to produce good classification method (Hastie and Tibshirani, 1998). However, it fails to outperform OVA because in the context of microarray, the small sample sizes cause the binary classifiers between each pair of classes to be unstable, thus reducing its generalization performance. PROB is the best decoding function for output-coding methods among the four. It significantly outperforms the other decodings on NCI60 data. On the other hand, PROB does not work well with OVA and AP. Further inspections reveal that the values of the sigmoid parameters for OVA and AP are sometimes exceptionally large (or small). This leads to a lot of errors for binary classifiers. The phenomenon is due to the small sample size which causes the class balance of positive and negative to be significantly biased. Although the regularization method proposed by Platt (1999) has been employed, the problem still appears. The convergence of logistic learning has been a long-standing

Table 5.5 Testing error rates for NCI60 data.

Gene = 20								
Decoding	**MLM**	**DIV**	**HAM**	**RHC**	**OVA**	**AP**	**KNN**	**C45**
HAMM	15.02±0.42	15.28±0.37	16.02±0.43	16.23±0.36	--	18.03±0.49	27.53±0.47	62.92±0.63
LOSS	13.82±0.38	14.62±0.37	14.42±0.41	14.82±0.33	--	--	--	--
INNR	13.82±0.38	14.63±0.37	14.37±0.41	14.82±0.33	14.68±0.41	--	--	--
PROB	**12.57±0.37**	14.18±0.37	14.08±0.48	14.75±0.35	33.83±1.26	24.68±0.99	--	--
Gene = 50								
Decoding	**MLM**	**DIV**	**HAM**	**RHC**	**OVA**	**AP**	**KNN**	**C45**
HAMM	13.18±0.44	14.43±0.45	15.13±0.42	15.73±0.39	--	19.78±0.46	32.97±0.49	67.35±0.68
LOSS	13.03±0.41	14.08±0.45	14.33±0.37	14.33±0.35	--	--	--	--
INNR	13.03±0.41	14.08±0.45	14.33±0.37	14.33±0.35	14.97±0.44	--	--	--
PROB	**11.77±0.36**	12.18±0.42	12.88±0.37	12.92±0.36	35.65±1.46	24.45±0.80	--	--
Gene = 100								
Decoding	**MLM**	**DIV**	**HAM**	**RHC**	**OVA**	**AP**	**KNN**	**C45**
HAMM	11.63±0.42	13.70±0.51	14.00±0.43	14.70±0.45	--	20.93±0.50	37.23±0.55	68.68±0.64
LOSS	11.82±0.44	12.37±0.44	13.05±0.41	13.05±0.42	--	--	--	--
INNR	11.82±0.44	12.37±0.44	13.05±0.41	13.05±0.42	14.20±0.56	--	--	--
PROB	**10.05±0.39**	10.38±0.37	10.75±0.44	11.42±0.37	30.07±1.29	26.43±0.86	--	--

Table 5.6 Testing error rates for GCM data.

Gene = 20								
Decoding	**MLM**	**DIV**	**HAM**	**RHC**	**OVA**	**AP**	**KNN**	**C45**
HAMM	26.05±0.20	26.44±0.21	27.24±0.20	27.85±0.21	--	29.28±0.23	34.24±0.19	44.67±0.32
LOSS	24.88±0.19	25.70±0.19	26.04±0.21	26.49±0.19	--	--	--	--
INNR	**24.74±0.19**	25.80±0.19	25.95±0.20	26.44±0.20	27.59±0.20	--	--	--
PROB	25.12±0.21	25.76±0.19	26.16±0.20	26.43±0.20	37.41±0.99	35.31±0.29	--	--
Gene = 50								
Decoding	**MLM**	**DIV**	**HAM**	**RHC**	**OVA**	**AP**	**KNN**	**C45**
HAMM	23.29±0.22	23.64±0.20	24.22±0.20	24.25±0.19	--	27.97±0.19	36.56±0.18	46.97±0.29
LOSS	22.42±0.22	22.94±0.20	23.24±0.20	23.48±0.20	--	--	--	--
INNR	22.34±0.22	22.92±0.20	23.23±0.20	23.46±0.20	23.66±0.20	--	--	--
PROB	**22.24±0.21**	22.41±0.19	23.04±0.20	23.13±0.19	35.19±1.05	34.34±0.22	--	--
Gene = 100								
Decoding	**MLM**	**DIV**	**HAM**	**RHC**	**OVA**	**AP**	**KNN**	**C45**
HAMM	22.71±0.23	24.12±0.20	23.85±0.20	24.15±0.22	--	29.06±0.22	38.20±0.19	45.42±0.30
LOSS	22.03±0.22	23.39±0.20	22.74±0.21	23.34±0.20	--	--	--	--
INNR	21.97±0.22	23.41±0.20	22.75±0.21	23.35±0.20	23.51±0.20	--	--	--
PROB	**21.68±0.23**	22.93±0.20	22.60±0.22	23.00±0.20	32.88±1.00	34.62±0.36	--	--

problem. A further investigation into the problem is beyond the scope of this work. Also, all decodings based on output values are better than HAMM. LOSS and INNR give almost identical results nonetheless.

All classifiers are influenced by gene selections. If more genes are used, less error rates are produced by the output-coding methods. However, this is not always the case for OVA and AP. AP even shows increasing error rates with more genes used on NCI60. This could be attributed to the relatively complex separating planes used by the output-coding methods. More information may be required to construct the boundaries induced by the codewords than the counterparts in OVA and AP. KNN and

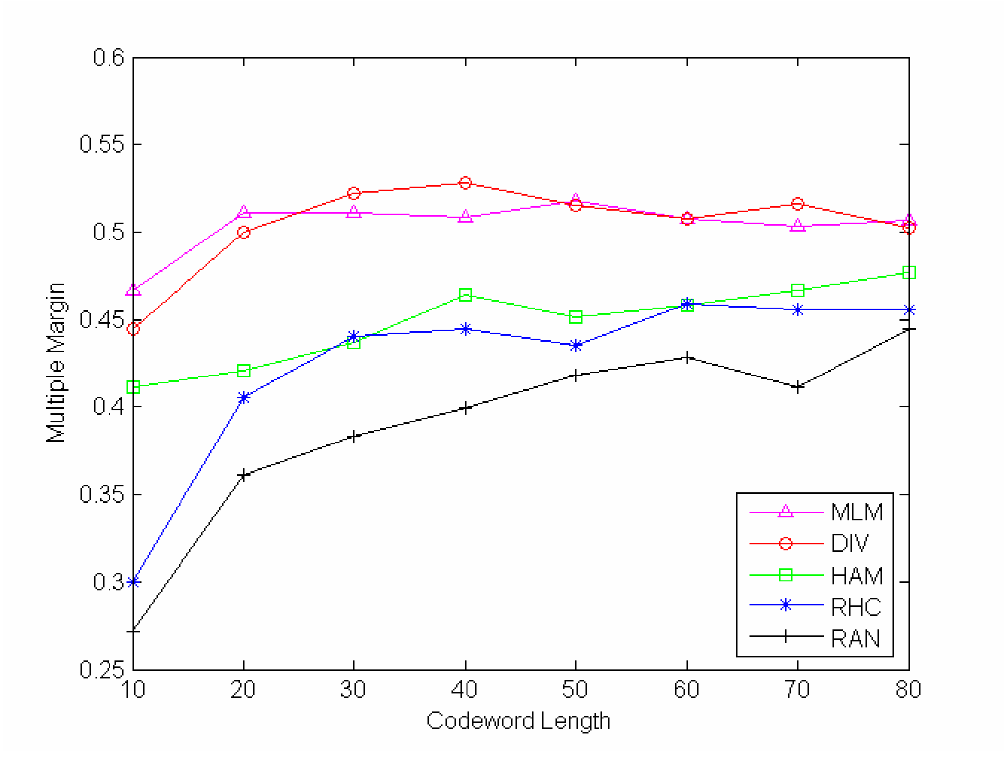
C45 are known to be very sensitive to the "curse of dimensionality" problem. It can be observed that their error rates increase with more genes used.

For those methods that are not used in this work, they are reviewed in a recently published paper that compares several multiclass classifiers (Statnikov *et al.*, 2005). On GCM, the best accuracy they have achieved is 76.60% using MC-SVM. On NCI60, the best accuracy they have achieved is less than 80% (this is cited from a graph, the exact number is not reported) using MC-SVM. They have also used NN and probabilistic neural network (PNN). However, these two classifiers perform worse than MC-SVM. It should be noticed that they have used 10-fold CV as an evaluation criterion. Because the parameter settings and evaluation methods are not exactly the same, it is not very reasonable to compare the two classifiers. Nevertheless, the output-coding methods have shown very promising performance against the existing methods in literature.

5.3.4 Codeword Length and Performance

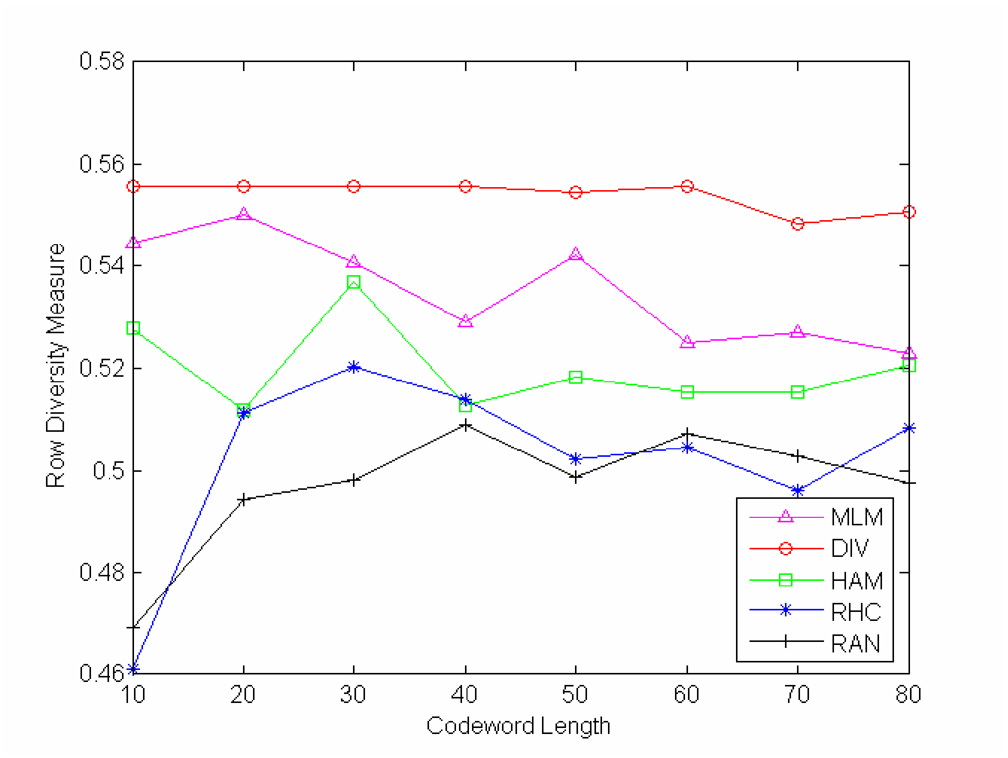
It is suggested by Windeatt and Ghaderi (2003) that the classification performance can usually be improved with longer codeword length. It may not be the case always. To investigate this matter in cancer classification, the codeword length is varied from 10 to 80 in a step of 10 for NCI60 data. Coding matrices are generated by MLM, DIV, HAM, RHC, and pure random (RAN) respectively. Thus, there are 40 matrices in total. The matrix measures of the 40 matrices versus codeword lengths are plotted in Figure 5.3(a)-(e). For each codeword length, MLM and RAN are selected to test the classifier error rates. HAMM decoding is used and 50 genes are selected for each class. Three-fold CVs are repeated for 100 times to avoid variations. The mean errors versus codeword lengths are plotted in Figure 5.3(f). It is obvious that all matrix measures of the GA generated coding matrices are improved against RAN. With a longer codeword length, M_r, D_r and H_r (row separation) increase automatically for RAN. This is because the number of possible codewords increases exponentially with the codeword length; thus the probability that two codewords have very small hamming-distance decreases significantly. However, D_c (column separation) decreases notably versus the codeword length. With k fixed, the possible number of binary classifiers is also fixed (Dietterich and Bakiri, 1995). Increasing the codeword length also increases the probability that two columns of the coding matrix are close to each other. This is not trivial since the binary classifiers are dependent. With a very small D_c, the average binary loss ε may be very large.

It is more straightforward to look at the mean errors of output-coding versus codeword length (Figure 5.3(f)). The RAN errors drop drastically from codeword length 10 to 30. However, from 30 on, no statistically significant improvements can be observed. The mean errors even increase from length 60 to 70. The MLM errors are smaller than that of RAN, especially on codeword length 10 and 20. This can be explained in an analytical way. Given a sample $\mathbf{x}$ belonging to a certain class, assume its margin is η. Then as long as $\eta > l\varepsilon$, $\mathbf{x}$ can be correctly classified. However, from Figure 5.3, we know that η does not increase linearly with l. Actually, it increases very fast when codeword length is short but slowly when codeword length becomes long. With a longer codeword length, the binary classifiers tend to be more dependent. This may cause ε to increase. Therefore, a longer codeword does not necessarily lead to statistically smaller error rate. Also, at long codeword length, the row and column separation of GA generated coding matrix and RAN become closer. Consequently, the MLM errors and RAN errors tend to be closer too. Similar results are observed on GCM data.



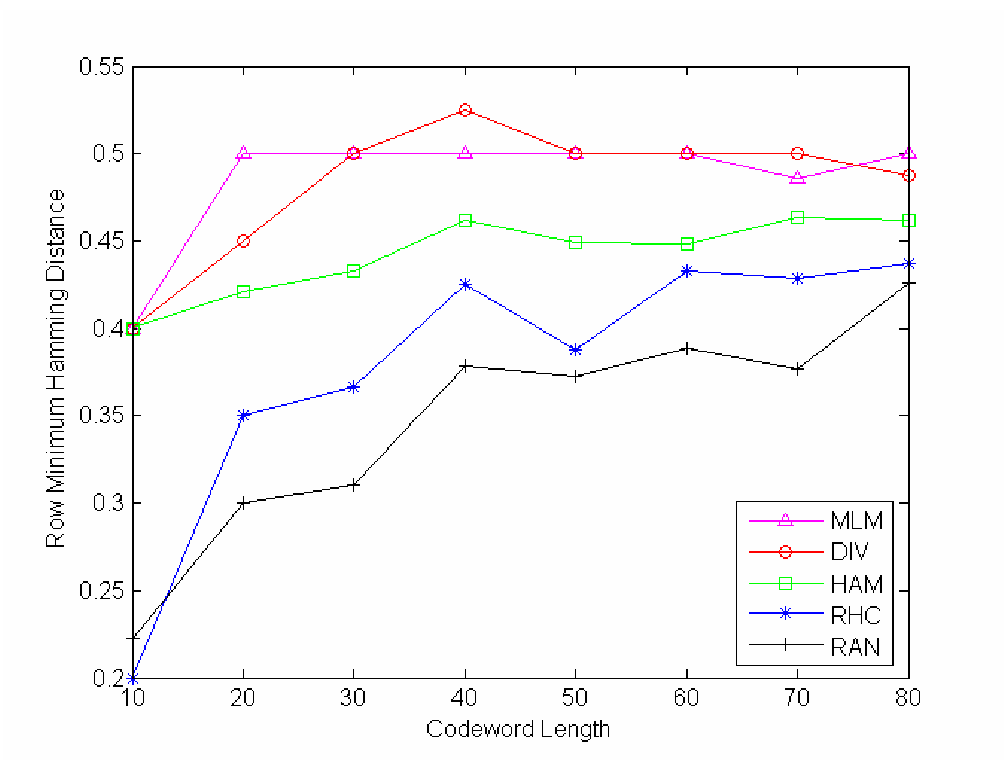
Multiple Margin
Codeword Length
MLM
DIV
HAM
RHC
RAN
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
10
20
30
40
50
60
70
80

(a)



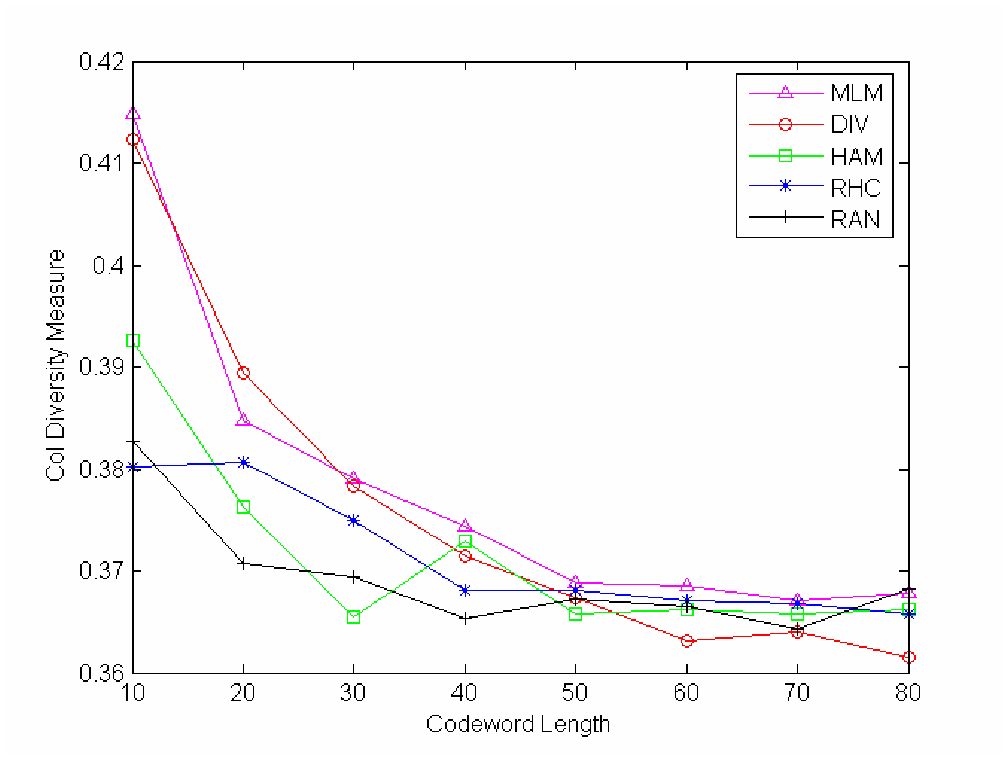
Row Diversity Measure
Codeword Length
MLM
DIV
HAM
RHC
RAN
0.58
0.56
0.54
0.52
0.5
0.48
0.46
10
20
30
40
50
60
70
80

(b)



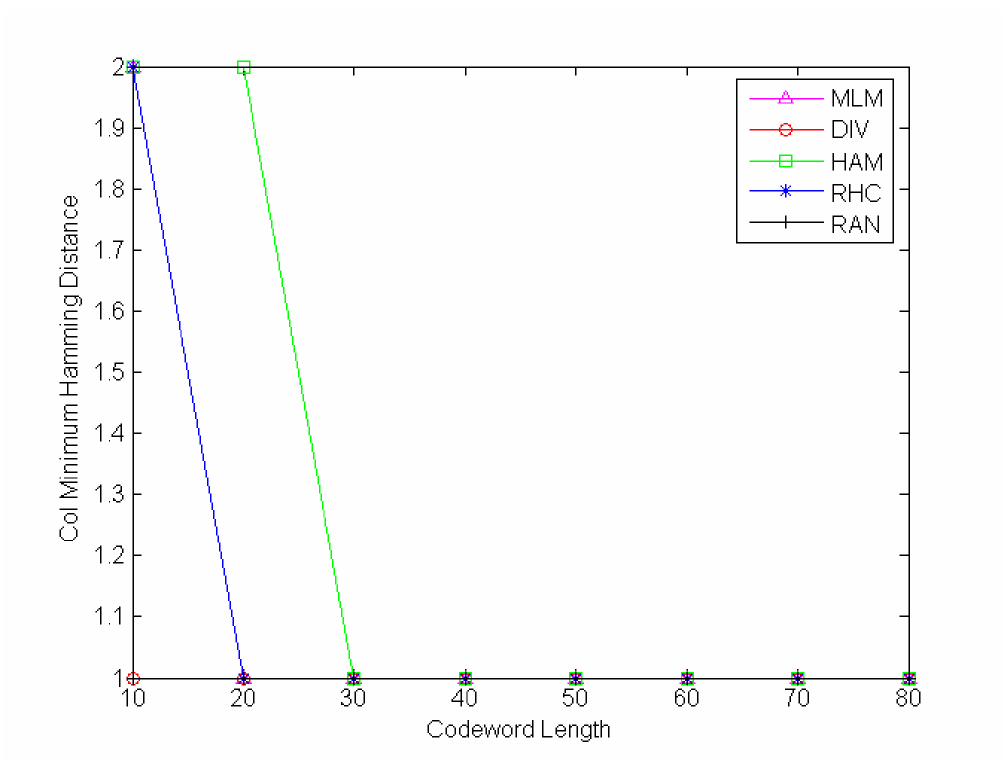
Row Minimum Hamming Distance
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
10
20
30
40
50
60
70
80
Codeword Length
MLM
DIV
HAM
RHC
RAN

(c)

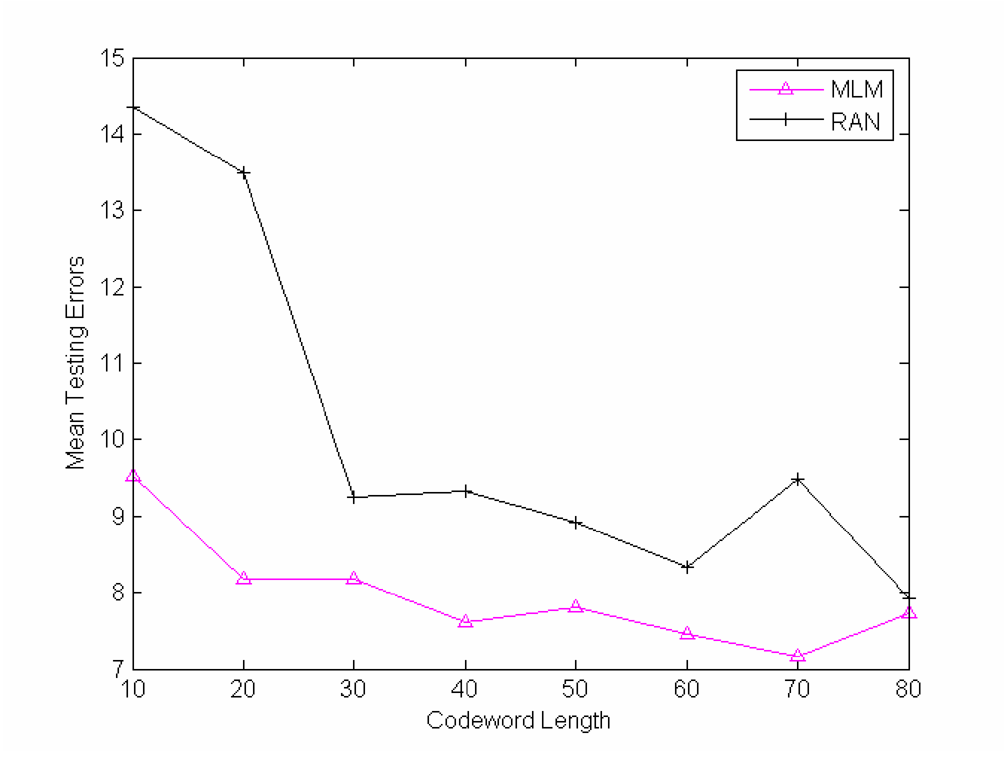


Col Diversity Measure
0.42
0.41
0.4
0.39
0.38
0.37
0.36
10
20
30
40
50
60
70
80
Codeword Length
MLM
DIV
HAM
RHC
RAN

(d)



(e)



(f)

Figure 5.3 Matrix measures and mean errors vs. codeword length. (a) M_r (b) D_r (c) H_r (d) D_c (e) H_c (f) Mean testing errors.

5.4 Discussions and Summary

Rather than using randomly generated matrix for output-coding classifiers, the GA has been shown to be a good method to search for optimal coding matrix given certain criterion. GA provides a simple yet effective method in the context of this work as convergence is usually guaranteed. Several measures of fitness have been compared. The new measure, namely MLM, seems to be better than the other two. Experiments have also shown that the decoding functions could have significant impact on the ensemble classifier performance.

It is found that a long codeword length does not necessarily give a statistically better performance. This may contradict common sense. The GA generated coding matrices are found to be superior to RAN matrices when codeword lengths are short. Based on the fact that a longer codeword length requires more computational efforts, short or average codewords generated by GA are recommended. Long codewords are only used when computational cost is not a concern.

In this work, only the properties of a coding matrix are considered in the optimization procedure. The training data and the binary classifier algorithms have not been taken into account. It should be noticed that the performance of output-coding is tightly influenced by the performance of binary classifiers on the training data. Feature selection and proper kernel parameter selection can potentially improve the binary classifiers.

Chapter 6. Contributions, Conclusions and Future Work

6.1 Contributions and Conclusions

The contributions are described as follows:

1. Two traditional regression methods have been introduced with a regularizer. Combined with dimension reduction, they can be effectively used for cancer classification. Several issues regarding the usage of the new methods like component selection, parameter tuning and feature selection have also been discussed.
2. The application of output-coding to enhance multiclass cancer classification has been presented. A systematic comparison of classification performance regarding multiple aspects of output-coding and other machine learning algorithms, namely KNN, C4.5 decision tree and neural networks, has been performed. Output-coding has been found to be superior.
3. Genetic algorithm (GA) has been used to jointly optimize the row separation and the column separation of a coding matrix. A new criterion based on a tighter error upper bound has been used to search for coding-matrix with better performance. Several issues regarding usage of GA like parameters setting, reproducibility and reliability have been discussed. In addition, the relationship between codeword length and classification accuracy has been investigated.

The conclusions are described as follows:

1. By using dimension reduction and regularization, the traditional regression methods can perform as well as or even better than some kernel classifiers like SVM. Only simple matrix inversion is required for the new methods. Therefore, the new methods can execute very efficiently because the sample size is usually small in the context of microarray.
2. Combining output-coding with binary classifiers is proved a very effective method for multiclass cancer classification. It is also a very flexible method. Several factors like coding-matrix, decoding function and feature selection can have significant impact on classification performance.
3. Genetic algorithm is found to be very effective in searching for coding matrix that is optimized in both row and column separation. There are several ways to define the row and column separation. It is preferred to apply a definition that is related to classification performance. “Multiple margins” is found to be a good criterion possibly because it gives a tighter error upper bound than minimum Hamming distance. The discussion of relationship between codeword length and classification performance reveals that binary classifier correlation is an important factor to classification performance. Although long codewords usually improve row separation, the binary classifiers become more dependent. This may increase the averaged binary errors. Therefore, long codewords do not necessarily lead to better performance. Tradeoff should be considered between performance and computational costs.

6.2 Future Work

Although binary classification has been extensively studied in machine learning, the application of machine learning techniques for both binary and multiclass cancer classification should continue to be important research issues for a few years. Accurate, reliable and cost-effective methods are still required to tackle the tasks of cancer classification. The new algorithms developed to solve cancer classification in this work can be applied to other fields like image processing or text classification. Therefore, they should be of interests to a much larger scientific community than the area of cancer classification. Some possible topics of future research are listed below:

1. Feature selection based on a small size sample could be unstable. Different classifiers may select different gene subsets. Same classifier with different parameters may also select different gene subsets. Re-sampling technique can be used to potentially solve this problem. Then how many bootstrap samples do we need to reduce the bias and variance for a certain threshold? Sometimes if the training sample size is small, even re-sampling cannot make satisfactory results. Then how many training samples do we need so that at least the re-sampling technique can work? These questions could be posed for both gene selection and prediction accuracies.
2. The classification performance is significantly influenced by the parameters of a classifier. If a kernel is used with the classifier, then the kernel parameter selection is another important issue. How to select the kernel as well as the set of parameters in an automatic and deterministic way? This would be of practical importance. Cross-validation can be used to assess whether the set of parameters is properly determined. However, it could be quite time-consuming and may not be suitable for large-scale parameter selection. It would also be interesting to select both parameters and genes simultaneously.

3. Prediction of continuous responses instead of categorical responses is also another area of active research (Park *et al.*, 2002; Kim *et al.*, 2005), e.g. prediction of patient survival time. Further, more, survival time is different from normal continuous responses. Failure times may not always be observed. This means that for some patients, failure occurs past a certain time but the exact time is not known. Poisson distribution may be used to model such kind of responses. Since the continuous responses are different from categorical responses, the formulation of objective functions should also be modified. However, the optimization principles beneath these prediction methods are generally the same.
4. Cancer classification and feature selection are not the final goals of cancer genomics. It is in vital demand to infer the gene regulatory networks from data. Some researchers have shown that Bayesian network is a powerful tool to infer the networks of gene interactions (Friedman *et al.*, 2000). However, to infer such a graph stably, the required sample size is much larger than the number of genes. Therefore, gene selection is also important consideration. Some research has to rely on synthetic data simply because the sample size of real data is not large enough.

Appendix A. Standard RLSC

We want to minimize

$$\sum_{i=1}^{n} \mathrm{V}(f(\mathbf{x}_i), y_i) + \lambda \|f\|_K^2 \tag{A.1}$$

given that V is the squared loss function and f has the form:

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i \, \mathrm{K}(\mathbf{x}, \mathbf{x}_i) + b \tag{A.2}$$

This can equally be written as the minimization function

$$\min J(\mathbf{c}, b) = \min_{\mathbf{c}, b} \frac{1}{2m} (\mathbf{Kc} + \mathbf{b} - \mathbf{y})^T (\mathbf{Kc} + \mathbf{b} - \mathbf{y}) + \frac{1}{2} \lambda \mathbf{c}^T \mathbf{Kc} \tag{A.3}$$

where J is objective function and $\mathbf{b}$ is a m-length vector with all entries equal to bias term b. Eliminating b first by taking partial derivative:

$$\begin{aligned} \frac{\partial J}{\partial b} &= \frac{1}{m} \mathbf{1}_m^T (\mathbf{Kc} + \mathbf{b} - \mathbf{y}) = 0 \\ \Rightarrow \quad b &= \bar{y} - \frac{1}{m} \mathbf{1}_m^T \mathbf{Kc} \end{aligned} \tag{A.4}$$

Because $\mathbf{K}$ is a positive definite matrix, J can be minimized by taking derivatives with respect to $\mathbf{c}$:

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{c}} &= \frac{1}{m} \mathbf{K} (\mathbf{Kc} + \mathbf{b} - \mathbf{y}) + \lambda \mathbf{Kc} = 0 \\ \Rightarrow \quad & \mathbf{Kc} + \mathbf{b} + m\lambda \mathbf{c} = \mathbf{y} \end{aligned} \tag{A.5}$$

substituting b in (A.5) with (A.4), we have

$$[(\mathbf{I} - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m^T) \mathbf{K} + m\lambda \mathbf{I}] \mathbf{c} = (\mathbf{I} - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m^T) \mathbf{y} \tag{A.6}$$

Appendix B. Dimension Reduction Based RLSC

Assuming $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k]$ is the matrix of mutually orthogonal components from KPLS or KPCA, i.e. $\mathbf{T}^T\mathbf{T} = \mathbf{I}$. We have the minimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2m}\left\|\mathbf{Tw} + \mathbf{b} - \mathbf{y}\right\|^2 + \frac{1}{2}\lambda\left\|\mathbf{w}\right\|^2 \tag{B.1}$$

Let $\mathbf{T}_1 = [\mathbf{1}_m, \mathbf{T}]$, $\mathbf{w}_1 = [b, \mathbf{w}]^T$ and $\mathbf{w}_2 = [0, \mathbf{w}]^T$, we equivalently have the minimization function

$$\min J(\mathbf{w}_1) = \min_{\mathbf{w}_1} \frac{1}{2m}(\mathbf{T}_1\mathbf{w}_1 - \mathbf{y})^T(\mathbf{T}_1\mathbf{w}_1 - \mathbf{y}) + \frac{1}{2}\lambda\mathbf{w}_2^T\mathbf{w}_2 \tag{B.2}$$

where J is the objective function. This can be achieved by taking partial derivative of $\mathbf{w}_1$

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{w}_1} &= \frac{1}{m}\mathbf{T}_1^T(\mathbf{T}_1\mathbf{w}_1 - \mathbf{y}) + \lambda\mathbf{w}_2 = 0 \\ \Rightarrow \quad & \mathbf{T}_1^T\mathbf{T}_1\mathbf{w}_1 + m\lambda\mathbf{w}_2 = \mathbf{T}_1^T\mathbf{y} \end{aligned} \tag{B.3}$$

Let $\sigma_i = \sum_{j=1}^{m} t_{ij}$ where t_{ij} is the j th element of $\mathbf{t}_i$, we have

$$\mathbf{T}_1^T\mathbf{T}_1 = \begin{bmatrix} \mathbf{1}_m^T \\ \mathbf{T} \end{bmatrix}\begin{bmatrix} \mathbf{1}_m & \mathbf{T} \end{bmatrix} = \begin{bmatrix} m & \sigma_1 & \cdots & \sigma_k \\ \sigma_1 & 1 & & \\ \vdots & & \ddots & \\ \sigma_k & & & 1 \end{bmatrix} \tag{B.4}$$

Therefore, from (B.3), we have

$$\begin{bmatrix} m & \sigma_1 & \dots & \sigma_k \\ \sigma_1 & 1 & & \\ \vdots & & \ddots & \\ \sigma_k & & & 1 \end{bmatrix} \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_k \end{bmatrix} + m\lambda \begin{bmatrix} 0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix} = \mathbf{T}_1^T \mathbf{y}$$

$$\Rightarrow \begin{bmatrix} m & \sigma_1 & \dots & \sigma_k \\ \sigma_1 & 1+m\lambda & & \\ \vdots & & \ddots & \\ \sigma_k & & & 1+m\lambda \end{bmatrix} \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_k \end{bmatrix} = \mathbf{T}_1^T \mathbf{y} \tag{B.5}$$

We can also solve (B.1) by eliminating b first and then taking partial derivative of $\mathbf{w}$ so that

$$[(1+m\lambda)\mathbf{I} - \frac{1}{m}\mathbf{T}^T \mathbf{1}_m \mathbf{1}_m^T \mathbf{T}]\mathbf{w} = \mathbf{T}^T(\mathbf{I} - \frac{1}{m}\mathbf{1}_m \mathbf{1}_m^T)\mathbf{y}$$
$$b = \bar{y} - \frac{1}{m}\mathbf{1}_m^T \mathbf{T}\mathbf{w} \tag{B.6}$$

However, (B.5) has a more simplified form and can solve $\mathbf{w}$ and b simultaneously.

List of Author's Publications

Journals

Shen, L. and Tan, E.C., "Dimension Reduction-Based Penalized Logistic Regression for Cancer Classification Using Microarray Data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 2, no. 2, pp. 166-175, 2005.

Shen, L. and Tan, E.C., "Combining Kernel Dimension-Reduction with Regularized Classifiers for Tissue Categorization," *Online Journal of Bioinformatics*, vol. 6, no. 1, pp. 91-98, 2005.

Shen, L. and Tan, E.C., "Reducing multiclass cancer classification to binary by output-coding and SVM," *Computaional Biology and Chemistry (Accepted for publication)*, 2005.

Shen, L. and Tan, E.C., "Optimized output-codes from genetic algorithm for multiclass cancer classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (Submitted for possible publication)*, 2005.

Conference Proceedings

Shen, L. and Tan, E.C., "A Generalized Output-Coding Scheme with SVM for Multiclass Microarray Classification," in *Fourth Asia-Pacific Bioinformatics Conference (APBC2006) (Accepted for publication)* Taipei, Taiwan, 13-16 February: 2006.

Shen, L. and Tan, E.C., "Nonlinear Kernel MSE Methods for Cancer Classification," in *First International Conference on Natural Computation 2005 (ICNC05),* LNCS 3610 ed. L. Wang, K. Chen, and Y. S. Ong, Eds. Changsha, China, 27-29 August: Springer-Verlag, 2005, pp. 975-984.

Shen, L. and Tan, E.C., "Cancer diagnosis by machine learning and microarray data," in *Seventh Annual NTU-SGH Symposium* Singapore, 11-12 August: 2005, p. 57.

Shen, L. and Tan, E.C., "PLS and SVD Based Penalized Logistic Regression for Cancer Classification Using Microarray Data," in *Third Asia-Pacific Bioinformatics Conference (APBC2005)*. P. Chen and L. Wong, Eds. Singapore, 17-21 January: Imperial College Press, 2005, pp. 219-228.

Shen, L. and Tan, E.C., "Gene Selection for Cancer Classification from Microarray Data using PLS-RLSC," in *First International Bioengineering Conference (IBEC2004)*. F. K. Fuss, S. L. Chia, S. S. Venkatraman, S. M. Krishnan, and B. Schmidt, Eds. Singapore, 8-10 September: 2004, pp. 73-76.

Shen, L. and Tan, E.C., "Efficient Algorithm for Gene Selection Using PLS-RLSC," in *Fourth International Conference on Bioinformatics of Genome Regulation and Structure (BGRS2004),* 2 ed Novosibirsk, Russia, 25-30 July: Russian Academy of Sciences, 2004, pp. 88-91.

References

Alizadeh, A.A., Eisen, M.B., Davis, R.E., Ma, C., Lossos, I.S., Rosenwald, A., Boldrick, J.C., Sabet, H., Tran, T., Yu, X., *et al.*, 2000, "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, pp. 503-511.

Allwein, E.L., Schapire, R.E., and Singer, Y., 2000, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113-141.

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., and Levine, A., 1999, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *PNAS*, vol. 96, pp. 6745-6750.

Antoniadis, A., Lambert-Lacroix, S., and Leblanc, F., 2003, "Effective dimension reduction methods for tumor classification using gene expression data," *Bioinformatics*, vol. 19, pp. 563-570.

Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., and Yakhini, Z., 2000, "Tissue Classification with Gene Expression Profiles," *Journal of Computational Biology*, vol. 7, no. 3/4, pp. 559-583.

Brown, P.O. and Botstein, D., 1999, "Exploring the new world of the genome with DNA microarrays," *Nature Genetics Supplement*, vol. 21, pp. 33-37.

Cho, S.B. and Won, H.H., 2003, "Machine Learning in DNA Microarray Analysis for Cancer Classification," in *Proceedings of The First Asia-Pacific Bioinformatics Conference (APBC2003)*. Y. Ping and P. Chen, Eds. Adelaide, Australia: Imperial College Press.

Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P.O., Herskowitz, I., 1998, "The transcriptional program of sporulation in budding yeast," *Science*, vol. 282, pp. 699-705.

Crammer, K. and Singer, Y., 2001, "On the learnability and design of output codes for multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265-292.

Debouck, C. and Goodfellow, P.N., 1999, "DNA microarrays in drug discovery and development," *Nature Genetics Supplement*, vol. 21, pp. 48-50.

DeRisi, J.L., Iyer, V.R., and Brown, P.O., 1997, "Exploring the metabolic and genetic control of gene expression on a genomic scale," *Science*, vol. 278, pp. 680-686.

Dettling, M. and Bühlmann, P., 2004, "Finding predictive gene groups from microarray data," *Journal of Multivariate Analysis*, vol. 90, pp. 106-131.

Dietterich, T.G. and Bakiri, G., 1995, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263-286.

Duda, R.O., Hart, P.E., and Stork, D.G., 2000, *Pattern Classification*, 2nd ed Wiley-Interscience.

Dudoit, S., Fridlyand, J., and Speed, T.P., 2002, "Comparison of Discriminant Methods for the Classification of Tumors Using Gene Expression Data," *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 77-87.

Duggan, D.J., Bittner, M., Chen, Y., Meltzerd, P., and Trend, J.M., 1999, "Expression profiling using cDNA microarrays," *Nature Genetics Supplement*, vol. 21, pp. 10-14.

de Jong, S., 1993, "SIMPLS: an alternative approach to partial least squares regression," *Chemometrics and Intelligent Laboratory Systems*, vol. 18, pp. 251-263.

Efron, B., 1975, "The efficiency of logistic regression compared to normal discriminant analysis," *Journal of the American Statistical Association*, vol. 70, no. 352, pp. 892-898.

Eilers, P.H.C., Boer, J.M., van Ommen, G.J.B., and van Houwelingen, H.C., 2001, "Classification of microarray data with penalized logistic regression," in *Proceedings of SPIE: progress in biomedical optics and imaging,* vol. 4266, pp. 187-198.

Fort, G. and Lacroix, S.L., 2005, "Classification using partial least squares with penalized logistic regression," *Bioinformatics*, vol. 21, pp. 1104-1111.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D., 2000, "Using Bayesian Networks to Analyze Expression Data," *Journal of Computational Biology*, vol. 7, pp. 601-620.

Fung, G. and Mangasarian, O.L., 2001, "Proximal support vector classifiers," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. San Francisco, California: ACM, pp. 77-86.

Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., and Haussler, D., 2000, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906-914.

Garthwaite, P.H., 1994, "An Interpretation of Partial Least Squares," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 122-127.

Ghosh, D., 2002, "Singular value decomposition regression models for classification of tumors from microarray experiments," *PSB*, vol. 98, pp. 18-29.

Ghosh, D., 2003, "Penalized Discriminant Methods for the Classification of Tumors from Gene Expression Data," *Biometrics*, vol. 59, pp. 992-1000.

Goldberg, D.E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison-Wesley.

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

Golub, G.H. and van Loan, C.F., 1996, *Matrix Computations* The Johns Hopkins University Press.

Gordon, G.J., Jensen, R.V., Hsiao, L., Gullans, S.R., Blumenstock, J.E., Ramaswamy, S., Richards, W.G., Sugarbaker, D.J. and Bueno, R., 2002, “Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer and Mesothelioma,” *Cancer Research*, vol. 62, pp. 4963-4967.

Grefenstette, J.J. and Baker, J.E., 1989, "How Genetic Algorithms Work: A Critical Look at Implicit Parallelism," in *Third International Conferenceon Genetic Algorithms*. J. D. Schaffer, Ed., pp. 20-27.

Gunn, S., 2001, "SVM MATLAB Toolbox", Available: http://www.isis.ecs.soton.ac.uk/resources/svminfo/.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V., 2002, "Gene selection for cancer classification using support vector machines," *Maching learning*, vol. 46, pp. 389-422.

Hastie, T. and Tibshirani, R., 1998, "Classification by pairwise coupling," *The Annuals of Statistics*, vol. 26, no. 2, pp. 451-471.

Hastie, T., Tibshirani, R., and Friedman, J., 2001, *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer.

Hoerl, A.E. and Kennard, R.W., 1970, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, pp. 55-67.

Huang, X. and Pan, W., 2003, "Linear regression and two-class classification with gene expression data," *Bioinformatics*, vol. 19, pp. 2072-2078.

Keller, A., Schummer, M., Hood, L., Ruzzo, W., 2000, "Bayesian classification of DNA array expression data," Technical Report, University of Washington.

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

Kim, H., Zhou, J.X., Morse, H.C., and Park, H., 2005, "A three-stage framework for gene expression data analysis by L1-norm support vector regression," *International Journal of Bioinformatics Research and Applications*, vol. 1, no. 1, pp. 51-62.

Kuncheva, L.I., 2005, "Using diversity measures for generating error-correcting output codes in classifier ensembles," *Pattern Recognition Letters*, vol. 26, pp. 83-90.

Lee, Y. and Lee, C.K., 2003, "Classification of multiple cancer types by multicategory support vector machines using gene expression data," *Bioinformatics*, vol. 19, pp. 1132-1139.

Li, J. and Liu, H., 2002, "Kent Ridge Biomedical Data Set Repository", Available: http://sdmc.lit.org.sg/GEDatasets/.

Li, T., Zhang, C., and Ogihara, M., 2004, "A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression," *Bioinformatics*, vol. 20, pp. 2429-2437.

Lilien, R.H., Farid, H., and Donald, B.R., 2003, "Probabilistic Disease Classification of Expression-Dependent Proteomic Data from Mass Spectrometry of Human Serum," *Journal of Computational Biology*, vol. 10, no. 6, pp. 925-946.

Lockhart, D.J. and Winzeler, E.A., 2000, "Genomics, gene expression and DNA arrays," *Nature*, vol. 405, pp. 827-836.

le Cessie, S. and van Houwelingen, J.C., 1992, "Ridge estimators in logistic regression," *Applied Statistics*, vol. 41, pp. 191-201.

Masulli, F. and Valentini, G., 2003, "Effectiveness of error correcting output coding methods in ensemble and monolithic learning machines," *Pattern Anal Applic*, vol. 6, pp. 285-300.

Mitchell, T.M., 1997, *Machine Learning*. New York: McGraw-Hill.

Nguyen, D.V. and Rocke, D.M., 2002, "Tumor classification by partial least squares using microarray gene expression data," *Bioinformatics*, vol. 18, pp. 39-50.

Park, P.J., Tian, L., and Kohane, I.S., 2002, "Linking gene expression data with patient survival times using partial least squares," *Bioinformatics*, vol. 18, p. s120-s127.

Passerini, A., Pontil, M., and Frasconi, P., 2004, "New Results on Error Correcting Output Codes of Kernel Machines," *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 45-54.

Peterson, C. and Ringnér, M., 2003, "Analyzing Tumor Gene Expression Profiles," *Artificial Intelligence in Medicine*, vol. 28, no. 1, pp. 59-74.

Petricoin III, E.F., Ardekani, A.M., Hitt, B.A., Levine, P.J., Fusaro, V.A., Steinberg, S.M., Mills, G.B., Simone, C., Fishman, D.A., Kohn, E.C., Liotta, L.A., 2002, “Use of proteomic patterns in serum to identify ovarian cancer,” *Lancet*, vol. 359, pp. 572-577.

Platt, J., 1999, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in *Advances in Large Margin Classifiers,* MIT Press.

Pomeroy, S.L., Tamayo, P., Gaasenbeek, M., Sturla, L.M., Angelo, M., McLaughlin, M.E., Kim, J.Y.H., Goumnerova, L.C., Black, P.M., Lau, C., Allen, J.C., Zagzag, D., Olson, J.M., Curran, T., Wetmore, C., Biegel, J.A., Poggio, T., Mukherjee, S., Rifkin, R., Califano, A., Stolovitzky, G., Louis, D.N., Mesirov, J.P., Lander, E.S. and Golub, T.R., 2002. "Prediction of central nervous system embryonal tumour outcome based on gene expression," *Nature*, vol. 415, pp. 436-442.

Press, S.J. and Wilson, S., 1978, "Choosing between logistic regression and discriminant analysis," *Journal of the American Statistical Association*, vol. 73, no. 364, pp. 699-705.

Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C.H., Angelo, M., Ladd, C., Reich, M., Latulippe, E., and Mesirov, J.P., 2001, "Multiclass cancer diagnosis using tumor gene expression signatures," *PNAS*, vol. 98, pp. 15149-15154.

Ricci, F. and Aha, D.W., 1998, "Error-Correcting Output Codes for Local Learners," in *Tenth European Conference on Machine Learning*.

Rifkin, R.M., 2002, "Everything Old Is New Again: A Fresh Look at Historical Approaches to Machine Learning." Massachusetts Institute of Technology.

Rifkin, R. and Klautau, A., 2004, "In Defense of One-Vs-All Classification," *Journal of Machine Learning Research*, vol. 5, pp. 101-141.

Rifkin, R., Mukherjee, S., Tamayo, P., Ramaswamy, S., Yeang, C.H., Angelo, M., Reich, M., Poggio, T., Lander, E.S., Golub, T.R., and Mesirov, J.P., 2003, "An Analytical Method for Multiclass Molecular Cancer Classification," *SIAM REVIEW*, vol. 45, no. 4, pp. 706-723.

Rosipal, R. and Trejo, L.J., 2001, "Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space," *Journal of Machine Learning Research*, vol. 2, pp. 97-123.

Schena, M., Shalon, D., Davis, R.W., and Brown, P.O., 1995, "Quantitative monitoring of gene expression patterns with a complementary DNA microarray," *Science*, vol. 270, pp. 467-470.

Schena, M., Shalon, D., Heller, R., Chai, A., Brown, P.O., Davis, R.W., 1996, "Parallel human genome analysis: microarray-based expression monitoring of 1000 genes," *Proc. Natl Acad. Sci. USA*, vol. 93, pp. 10614-10619.

Schimek, M.G., 2003, "Penalized logistic regression in gene expression analysis", Available: http://www.quantlet.org/hizirjsp/schimek/schimek.pdf.

Schölkopf, B. and Smola, A., 2002, *Learning with Kernels* MIT Press.

Schölkopf, B., Smola, A., and Müller, K., 1998, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, pp. 1299-1319.

Schwaighofer, A., 2001, "SVM MATLAB Toolbox", Available: http://www.cis. tugraz.at/igi/aschwaig/svm_v251.tar.gz.

Shen, L. and Tan, E.C., 2005a, "Dimension Reduction-Based Penalized Logistic Regression for Cancer Classification Using Microarray Data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 2, no. 2, pp. 166-175.

Shen, L. and Tan, E.C., 2005b, "Reducing multiclass cancer classification to binary by output-coding and SVM," *Computaional Biology and Chemistry (Accepted for publication)*.

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

Statnikov, A., Aliferis, C.F., Tsamardinos, I., Hardin, D., and Levy, S., 2005, "A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis," *Bioinformatics*, vol. 21, pp. 631-643.

Stephanopoulos, G., Hwang, D., Schmitt, W.A., Misra, J., and Stephanopoulos, G., 2002, "Mapping physiological states from microarray expression measurements," *Bioinformatics*, vol. 18, pp. 1054-1063.

Suykens, J.A.K. and Vandewalle, J., 1999, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, pp. 293-300.

Tan, A.C. and Gilbert, D., 2003, "Ensemble machine learning on gene expression data for cancer classification," *Applied Bioinformatics*, vol. 2, p. s75-s83.

Valentini, G., 2002, "Gene expression data analysis of human lymphoma using support vector machines and output coding ensembles," *Artificial Intelligence in Medicine*, vol. 26, pp. 281-304.

Van't Veer, L.J., Dai, H.Y., Van De Vijver, M.J., He, Y.D., Hart, A.A.M., Mao, M., Peterse, H.L., Van Der Kooy, K., Marton, M.J., Witteveen, A.T., Schreiber, G.J., Kerkhoven, R.M., Roberts, C., Linsley, P.S., Bernards, R., and Friend, S.H., 2002, "Gene expression profiling predicts clinical outcome of breast cancer," *Nature*, vol. 415, pp. 530-536.

Vapnik, V., 1998, *Statistical Learning Theory*. New York: John Wiley and Sons, Inc.

Weaver, R.F., 2004, *Molecular Biology*, 3 ed McGraw-Hill.

West, M., Blanchette, C., Dressman, H., Huang, E., Ishida, S., Spang, R., Zuzan, H., Olson, J.A., Marks, J.R., and Nevins, J.R., 2001, "Predicting the clinical status of human breast cancer by using gene expression profiles," *PNAS*, vol. 98, no. 20, pp. 11462-11467.

Wegelin, J.A., 2000, "A survey of Partial Least Squares (PLS) methods, with emphasis on the two-block case," Department of Statistics, University of Washington, Seattle.

Windeatt, T. and Ghaderi, R., 2003, "Coding and decoding strategies for multi-class learning problems," *Information Fusion*, vol. 4, pp. 11-21.

Wodicka, L., Dong, H., Mittmann, M., Ho, M.H., and Lockhart, D.J., 1997, "Genome-wide expression monitoring in *Saccharomyces cerevisiae*," *Nat. Biotechnol.*, vol. 15, pp. 1359-1367.

Wold, H., 1966, "Estimation of principal components and related models by iterative least squares," in *Multivariate Analysis*. P. R. Krishnaiah, Ed. New York: Academic Press, pp. 391-420.

Yeang, C.H., Ramaswamy, S., Tamayo, P., Mukherjee, S., Rifkin, R.M., Angelo, M., Reich, M., Lander, R., Mesirov, J., and Golub, T.R., 2001, "Molecular classification of multiple tumor types," *Bioinformatics*, vol. 17, p. s316-s322.

Yeoh, E.J., Ross, M.E., Shurtleff, S.A., Williams, W.K., Rami Mahfouz, D.P., Behm, F.G., Raimondi, S.C., Relling, M.V., Patel, A., and Cheng, C., 2002, "Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling," *Cancer Cell*, vol. 1, pp. 133-143.

Yu, H., 2004, "Data Mining Via Support Vector Machines: Scalability, Applicability, and Interpretability," Doctoral Thesis.

Zhang, H., Yu, C., Singer, B., Xiong, M., 2001, "Recursive partitioning for tumor classification with gene expression microarray data," *Proc. Natl. Acad. Sci. USA,* vol. 98, pp. 6730–6735.

Zhu, J. and Hastie, T., 2004, "Classification of gene microarrays by penalized logistic regression," *Biostatistics*, vol. 16, pp. 427-443.