University of Alberta

Statistical Pattern Recognition Methods for Diagnosis of Cancer Using Gene Expression Data

by

Elsa Naser ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

in

Statistics

Department of Mathematical and Statistical Sciences

Edmonton, Alberta

Spring 2002

0-612-69742-8

University of Alberta

Library Release Form

Name of Author:	Elsa Naser
Title of Thesis:	Statistical Pattern Recognition Methods for Diagnosis of Cancer Using Gene Expression Data
Degree:	Master of Science
Year this Degree Granted:	2002

Elsa Naser

10630-110 ST, Suite 22,
Edmonton, AB, T5H 3C8
Canada

Date: 22/03/2002

UNIVERSITY OF ALBERTA

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Statistical Pattern Recognition Methods for Diagnosis of Cancer Using Gene Expression Data** submitted by **Elsa Naser** in partial fulfillment of the requirements for the degree of **Master of Science** in Statistics.

Dr. N. G. N. Prasad (Chair)

Peter Hoop

Dr. Peter Hooper (Supervisor)

Russell Greiner

Dr. Russ Greiner (Computing Science)

January 7, 2002

Abstract

Although cancer classification is usually accurate, it remains imperfect and errors do occur. The new technologies, called microarray technology, promise to monitor the whole genome on a single chip so that we can have a better picture of the interactions among thousands of genes simultaneously. Recent studies in cancer research are using this technology to conduct genome-scale characterizations of gene expression in human tumors. Most published papers on tumor classification have applied a single technique to a single expression data set. It is difficult however to assess the merits of each technique in the absence of comparative study.

In this thesis, we compared the performance of Reference Point Logistic regression, with the two classifiers that are found to perform well in Dudoit et al. (2000): linear discriminant analysis and the k-nearest neighbor classifier. Quadratic discriminant analysis was also included in our comparison to see its performance compared with linear discriminant analysis.

Acknowledgements

I am really grateful to my supervisor, Prof. P.M. Hooper, for leading me to the research field of this thesis, and his enormous support, crucial comments and suggestions, encouragement and kindness throughout the preparation of this thesis. Without his guidance, this thesis would have been impossible.

I also express my gratitude to my thesis committee members, Dr. N.G. Prasad and Dr. Russell Greiner, for their time spent in reading my thesis and giving me invaluable suggestion.

I would also like to thank all my friends in particular Meseret, Meserach, Getu, Ergete and Getye for their priceless help in the preparation of this thesis.

Thanks are also due to the Department of Mathematical and Statistical sciences for providing me the opportunity to study here and for the financial support; and to all faculty members who taught me courses. I also acknowledge the assistance of department secretaries.

Finally, my special thank you goes to my husband, Mohamud, for his love and support throughout the program.

Table of Contents

Pages

List of Tables

List of Figures

Notation

The notation used in this paper is listed below. A lower case letter in bold face denotes a vector.

p	Number of features (genes)
J	Number of classes (cancer type)
n	Total number of cases (tissue sample)
n_y	Number of cases in the yth class, $y=1,\cdots,J$
$p(y)$	Prior probability of class y
x_{ij}	Gene expression of the *jth* gene for the *ith* case
$\boldsymbol{x}_i=(x_{i1},\cdots,x_{ip})$	Gene expression profile vector for the *ith* case
$y_i \in \{1,\cdots,J\}$	Class label for the *ith* case
$f(\boldsymbol{x} \mid y)$	Class conditional density of $\boldsymbol{x}$ given y
$p(y \mid \boldsymbol{x})$	Conditional density of y given $\boldsymbol{x}$.
$\boldsymbol{\mu}_y=(\mu_{y1},\cdots,\mu_{1p})$	Mean vector for class y, $y=1,\cdots,J$
$\bar{\boldsymbol{x}}_y=(\bar{x}_{y1},\cdots,\bar{x}_{yp})$	Sample mean vector for class y, $y=1,\cdots,J$
$\boldsymbol{\Sigma}_y$	Covariance matrix for class y, $y=1,\cdots,J$
B	Number of times we rerun the classifiers

CHAPTER 1

INTRODUCTION

Despite the variety of clinical, morphological and molecular parameters used to classify human malignancies today, patients receiving the same diagnosis can have markedly different clinical courses and treatment responses. Current clinical practice involves an experienced hematopathologist's interpretation of the tumor's morphology, histochemistry, immunophenotyping, and cytogenetic analysis, each performed in a separate, highly specialized laboratory. Although cancer classification is usually accurate, it remains imperfect and errors do occur.

The new technologies, called complementary DNA (cDNA) microarray and high-density oligonucleotide chips, promise to monitor the whole genome on a single chip so that we can have a better picture of the interactions among tens of thousands of genes simultaneously. Recent studies in cancer research are using this technology to conduct genome-scale characterizations of gene expression in human tumors, with the goal of developing improved and higher resolution methods for classifying tumors, which in turn should lead to more specific and effective treatment strategies.

Some of the statistical problems associated with cancer diagnosis are: defining previously unrecognized tumor subtypes – *unsuperpervised learning*, the assignment of particular tumor samples to already-defined classes – *supervised learning*, and the identification of "marker" genes that characterize the different tumor classes – *variable selection*. This thesis focuses on the second problem, the classification of cancer using gene expression data.

Both the supervised and unsupervised learning methods have been employed in some of the recent cancer studies using gene expression data. Golub, Slonim, Tamayo, Huard, Gaasenbeek, Mesirov, Coller, Loh, Downing, Caligiuri, Bloomfield and Lander (1999) looked into both cluster analysis and discriminant analysis of tumors using gene expression data. For cluster analysis, Self Organizing Maps (SOM) was applied to the gene expression data and the cancer types revealed by this method were compared to known classes. For discriminant analysis, Golub et al. (1999) proposed a weighted gene-voting scheme.

Alizadeh, Eisen, Davis, Ma, Lossos, Rosenwald, Boldrick, Sabat, Tran, Yu, Powell, Yang, Marti, Moore, Hudson, Lu, Lewis, Tibshrani, Sherlock, Chan, Greiner, Weisenburger, Armitage, Warnke, Levy, Wilson, Grevel, Byrd, Botstein, Brown and Staudt (2000) applied a hierarchical clustering algorithm, to group both the genes and tumors, on gene expression data for the three most prevalent adult lymphoid malignancies: diffuse large B cell lymphoma (DLBCL), follicular lymphoma (FL), and chronic lymphocytic leukemia (CLL). Two types of B cell malignancies that are not recognized by the current classification were identified.

Ross, Scherf, Eisen, Perou, Rees, Spellman, Iyer, Jeffrey, Van de Rijn, Waltham, Pergamenschikov, Lee, Lashkari, Shalon, Myers, Weinstein, Botstein, and Brown (2000) used cDNA microarrays to explore variation among gene expression profiles in 60 human cancer cell lines (NCI60) derived from tumors from a variety of tissue and organs. The authors used a hierarchical clustering algorithm to group cell lines with similar gene expression and the result of analysis revealed a correspondence with the ostensible origins of the tumors from which the cell lines were derived. The same clustering algorithm was employed to group genes whose expression level varied among the 60 cell lines.

Most published papers on tumor classification have applied a single technique to a single expression data set. It is difficult however to assess the merits of each technique in the absence of comparative study. Dudoit, Fridlyand and Speed (2000) compared the performance of different discriminant methods for classification of tumors based on gene expression data. Simple classifiers such as linear discriminant analysis and nearest neighbors performed well compared to the more sophisticated methods such as aggregated classification trees.

In this thesis, the gene expression datasets used in Dudoit et al. (2000) are used to compare the performance of Reference Point Logistic (RPL) regression, which was introduced by Hooper (1999, 2001), with the two classifiers that performed well in Dudoit et al. (2000): linear discriminant analysis (based on the normal distribution) and the k-nearest neighbor classifier (a nonparametric classifier). Quadratic discriminant analysis is also included in our comparison to compare its performance with linear discriminant analysis.

RPL had been applied to ten StatLog data sets (Michie, Spiegelhalter, and Taylor, 1994) and it performed well in comparison with 22 other classification methods. Detailed results are given in Hopper (2001).

RPL regression models $p(y|x)$ according to proximity between x and reference points for each class. The numbers of reference points assigned to each class control the complexity of the model. These numbers can be selected empirically using cross-validated risk estimates. When a single reference point is assigned to each class, RPL regression is equivalent to multinomial logistic regression. When applying RPL to the microarray data sets, we found that choosing a single reference point per class gave the best result. Thus our evaluation of RPL can be viewed as an evaluation of logistic regression.

Sometimes, the choice of more than one reference point per class provides a good result. For example, in the problem of identifying functional sites at the boundaries of protein coding regions in genomic DNA (Hooper et al., 2001) a more complex model gave a better result.

An unusual feature of gene expression data is the very large number of variables (genes) as compared to the number of cases (tumor samples). The publicly available datasets currently contain gene expression data for 5,000 – 20,000 genes on fewer than 100 tumor samples.

To reduce the dimensionality of the data, Ross et al. (2000) used two different subsets of genes: 1,161 and 6,831 genes out of a total of 9,703 genes. The 1,161 cDNAs were those with transcript levels that varied by at least sevenfold $(\log_2(ratio) > 2.8)$ relative to the reference pool in at least 4 of 60 cell lines. This effectively selects genes with greatest variation in expression level across the 60 cell lines, and therefore highlights those gene expression patterns that best distinguished the cell lines from one another.

The 6,831 cDNAs were those with a minimum fluorescence signal intensity of approximately 0.4% of the dynamic range above background in the reference channel in each of the six hybridizations used to establish reproducibility. This effectively selected those genes that provided the most reliable ratio measurements and therefore identified a subset of genes useful for exploring patterns comprised of moderately large variation in expression across the 60 cell line was of moderate magnitude.

To reduce the dimensionality of the data, Dudoit et al. (2000) performed a preliminary selection of genes on the basis of the ratio of their between groups to within groups sum of squares.

To reduce the dimensionality of the data set used in this thesis, we used a k-means clustering criterion to group together genes with "similar" patterns of expression. The gene centroid vectors of each cluster are then used as potential features to develop the classifiers.

This thesis is organized as follows. We begin in Chapter 2 by briefly discussing the biological background and technology of cDNA microarrays. Chapter 3 discusses some of the statistical methods recently used in the study of human cancer using gene expression data. Chapter 4 describes the data set used in this paper, along with preliminary data processing steps and dimensionality reduction. The classification methods considered in the paper are discussed in Chapter 5. Finally, Chapter 6 summarizes our findings and outlines open questions.

CHAPTER 2

BIOLOGICAL BACKGROUND ON cDNA MICROARRAYS

A gene is a discrete sequence of DNA encoding a particular protein, the ultimate expression of the genetic information. DNA consists of two associated polynucleotide strands that wind together in a helical fashion, often described as a double helix. Each of the nucleotides is composed of deoxyribose sugar, a phosphate group, and one of the four nitrogen bases: adenine (A), thymine (T), guanine (G) and cytosine(C). Phosphate and sugars of adjacent nucleotides link to form a long polymer (a large molecule containing repeating units) and the two strands of DNA are linked by complementary pairs of nitrogen bases- A always paired with T, and G always paired with C. The right panel of Figure 1 on page 7 shows the basic structure of DNA.

The entire process that takes the information contained in genes on DNA and turns that information into proteins, which in turn determine the properties of cells, is called gene expression. In eukaryotic species (multi-cellular organisms), the DNA sequences coding for proteins called exons are interrupted by stretches of non-coding DNA called introns.

We can summarize the three steps in gene expression as follows. First, the DNA that includes all the exons and introns of the gene is transcribed to produce nRNA (nuclear RNA); a single stranded complementary copy of the gene, with the base uracil (U) replacing thymine (T). See the left panel of Figure1 for basic structure of RNA. In the second step, introns are spliced out from nRNA to produce mRNA (messenger RNA). Figure 2 displays the transcription of DNA to mRNA.

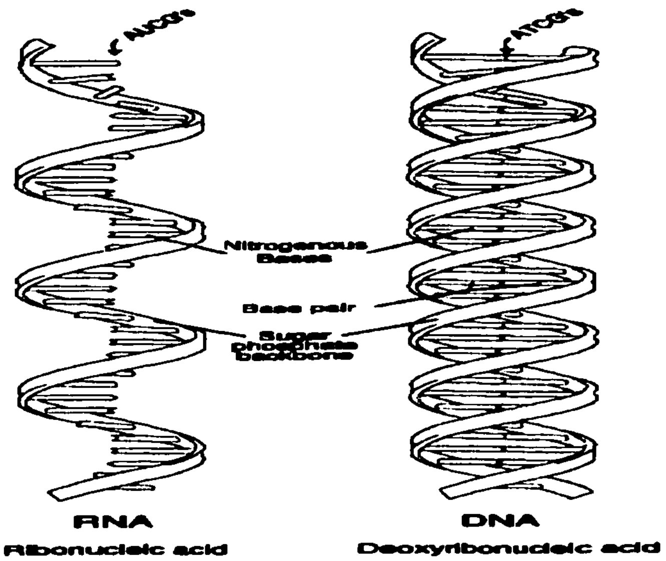

Figure 1: Structure of RNA and DNA.

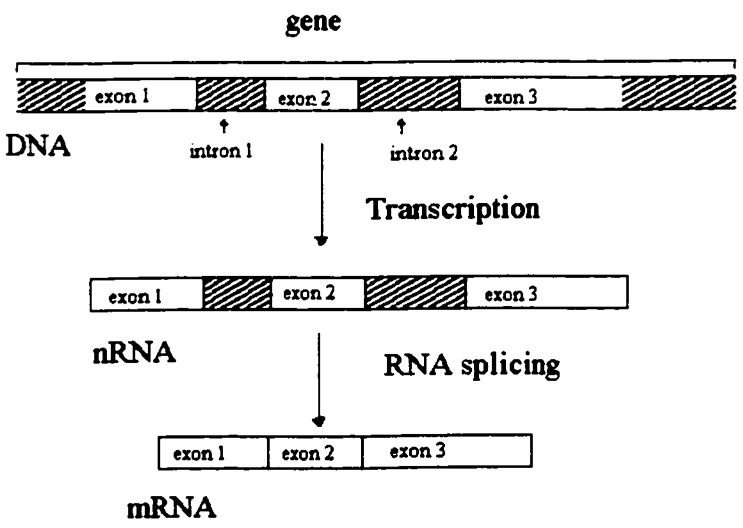

Figure 2: The transcription of DNA to mRNA in Eukaryotic species.

Both figures 1 and 2 are obtained from Access Excellence, National health Museum (1999): http://www.accessexcellence.org

Finally, the mRNA is transported to the cytoplasm for translation into protein. For an introduction to basic genetics, see Hartl (1991).

It is widely believed that some genetic diseases are caused by genes that are inappropriately transcribed (either too much or too little) or that are missing completely. Such defects are especially common in cancers.

Because mRNA is an exact copy of the DNA coding regions, mRNA analysis can sensitively reflect the type and state of the cell. Microarray technology is one of several developing approaches to comparatively analyze genome-wide patterns of mRNA expression. To prepare mRNA for use in a microarray assay, it must be purified from total cellular contents. Captured mRNAs are still difficult to work with because they are prone to being destroyed. The environment is full of RNA-digesting enzymes (there are some on your fingers, your keyboard, your mouse, and every other exposed surface around you right now), so free RNA is quickly degraded. To prevent the experimental samples from being lost, the RNA is reverse-transcribed back into more stable single stranded DNA form. The products of this reaction are called complementary DNA (cDNA) because their sequences are the complements of the original mRNA sequences.

DNA microarrays or DNA chips consist of large numbers of specific oligonucleotide or cDNA sequences called probes, each corresponding to a different gene, affixed to a solid surface at very precise locations. When an array chip is hybridized to labeled cDNA derived from a particular tissue of interest, it yields simultaneous measurements of the mRNA levels in the sample for each gene represented on the chip.

The data often contain technical noises that can be introduced at a number of different stages, such as production of the DNA array, preparation of samples, hybridization between cDNA and array, signal analysis and extraction of the hybridization results. Schena et al., (1995) reduced some of this noise by simultaneously hybridizing both a test and reference sample to an array, each labeled with a different color fluorescent dye. An overview to DNA microarray technology with its application can be found in Schena (1999).

CHAPTER 3

SELECTIVE REVIEW OF LITERATURE

This chapter will review some of the statistical methods that have been used to study human cancer using gene expression data.

3.1 Supervised Harvesting of Expression Trees

Hastie, Tibshirani, Botstein and Brown (2000) proposed a new method, named tree harvesting, for supervised learning from gene expression data. The technique starts with a hierarchical clustering of genes and then considers the average expression profiles and their products (to capture interaction effects) from all of the clusters in the resulting dendrogram as potential inputs into the prediction model.

Suppose we have gene expression data x_{ij} for genes $j = 1,2,\ldots,p$ and tissue (cases) $i = 1,2,\cdots,n$, and a response measure y_i for each case. The response can be quantitative, for example survival time, or categorical, in which case y_i is assumed to take values in $\{1,2,\cdots,J\}$.

Let $C_g \subseteq /1,2,\cdots,p\}$ denote a cluster of n_g genes. The corresponding average expression profile is given by:

$$\bar{\boldsymbol{x}}_g = \frac{1}{n_g} \sum_{j \in C_g} \boldsymbol{x}_j$$

where $\boldsymbol{x}_j \in \Re^n$ is gene expression profile vector for gene j.

Starting with p genes, a given hierarchical clustering algorithm produces $2p-1$ such clusters, including the individual genes themselves. To facilitate construction of the interaction model, each x_{ij} is translated to have a minimum value 0 over the cases:

$$x^*_{ij} = x_{ij} - min\{\, x_{kj} : k = 1, 2, \cdots, n \,\}$$

Let $\bar{x}^*_g$ denote the average expression profile for cluster g using these translated values.

For a categorical response, the most commonly used model, multiple logistic regression has the form:

$$log \frac{p(Y = y \mid x)}{p(Y = J \mid x)} = \beta_{y0} + \sum_g \beta_{y.g} \bar{x}^*_g + \sum_{g'} \sum_g \beta_{y.gg'} \bar{x}^*_g \bar{x}^*_{g'} + \cdots$$

The user can put an explicit limit on the order of the interaction allowed in the model. In fact in the examples considered in the study, the products are allowed to be pair-wise products.

The model is then developed in a forward stepwise manner as follows: Initially the only term in the model M is the constant function 1. The candidate terms C consists of all the $2p-1$ average expression profiles $\bar{x}^*_g$. At each stage, all products consisting of a term in M and a term in C will be considered and the term that most improves the fit of the model in terms of a score statistic S will be included in the model M. This will be continued until some maximum number of terms m has been added to the model.

At stage 2 of the procedure, selection is biased towards larger clusters, as most of the clusters considered in the harvest procedure are subsets of other clusters. Hence if an average $\bar{x}_g^*$ is found to most improve the fit of the model, it is likely that the average expression profile of some larger cluster, perhaps containing the chosen cluster, does nearly as well as $\bar{x}_g^*$. If the score for cluster g is S_g, the algorithm will choose the largest cluster g' whose score $S_{g'}$ is within a factor $(1-\alpha)$ of the best; that is, satisfying $S_{g'} \geq (1-\alpha) S_g$. The parameter α is set by the user. In the study, they set $\alpha = 0.10$.

As to the model size selection, 10-fold cross-validation was used in the following manner:

- Having built a harvest model with some large number of terms m, carry out backward deletion, at each stage dropping the term that causes the smallest increase in the sum of squares. This is continued until the model contains only the constant term. This gives a sequence of models with terms $1,2,\cdots,m$.
- And then 10-fold cross validation is used to choose the best model size.

The tree harvesting procedure was illustrated in two real examples: survival time of lymphoma patients and NCI60 human tumor data. A simulation study was also carried out to assess how well the tree harvesting discovers "true" structure.

5.1 Comparison of Discriminant Methods for the Classification of Tumors using Gene Expression Data

Dudoit et al. (2000) compared the performance of different discrimination methods for classification of tumors based on gene expression data. These methods include: Fisher linear discriminant analysis, maximum likelihood discriminant rules, nearest neighbor classifier, and classification trees. The methods were applied to three datasets: NCI60, lymphoma and leukemia datasets.

To reduce the dimensionality of the data, the authors performed a preliminary selection of genes on the basis of the ratio of their between groups to within groups sum of squares. For gene j, this ratio is:

$$\frac{BSS(j)}{WSS(j)} = \frac{\sum_y \sum_i I(y_i = y)(\bar{x}_{yj} - \bar{x}_{.j})^2}{\sum_y \sum_i I(y_i = y)(x_{ij} - \bar{x}_{yj})^2}$$

where $\bar{x}_{.j}$ denotes the average expression level of gene j across all cases and $\bar{x}_{yj}$ denotes the average expression level of gene j across cases belonging to class y.

The authors selected the d genes with the largest BSS/WSS : $d = 50$ for the lymphoma dataset, $d = 40$ for the leukemia dataset, and $d = 30$ for the NCI60 dataset. The effect of increasing d to 200 or decreasing it to 10 was examined.

When the class conditional densities are known, the maximum likelihood discriminant rule assigns an observation vector $\boldsymbol{x}$ to the class y maximizing $f(\boldsymbol{x} \mid y)$. For a multivariate normal class conditional density, the rule assigns a new observation vector $\boldsymbol{x}$ to a class y minimizing

$$(x-\mu_y)^T \Sigma_y^{-1} (x-\mu_y) + log\, |\Sigma_y| \qquad (3.1)$$

where $|\Sigma_y|$ is the determinan t of Σ_y

Two special cases of (3.1) were considered in the study. When each of the class densities has diagonal covariance matrix $\Sigma_y = diag(\sigma_{y1}^2,\cdots,\sigma_{yp}^2)$ rule (3.1) assigns x to class y minimizing

$$\sum_{j=1}^{p} \left\{ \frac{(x_j - \mu_{yj})^2}{\sigma_{yj}^2} + log\, \sigma_{yj}^2 \right\} \qquad (3.2)$$

Moreover, when the class densities have a common diagonal covariance matrix $\Sigma = diag(\sigma_1^2,\cdots,\sigma_p^2)$, rule (3.1) assigns x to class y minimizing

$$\sum_{j=1}^{p} \left\{ \frac{(x_j - \mu_{yj})^2}{\sigma_j^2} \right\} \qquad (3.3)$$

The authors referred the rules (3.2) and (3.3) as a diagonal quadratic and linear discriminant rules respectively.

For the corresponding sample ML discriminant rules, μ_y and Σ_y are estimated from the learning set by $\bar{x}_y = (\bar{x}_{y1},\cdots,\bar{x}_{yp})$ and $\hat{\Sigma}_y = diag(S_{y1}^2,\cdots,S_{yp}^2)$ respectively as follows:

$$\bar{x}_{yj} = \frac{1}{n_y} \sum_{i=1}^{n} I(y_i = y) x_{ij}$$

$$S_{yj}^2 = \frac{\sum_{i=1}^{n} I(y_i = y)(x_{ij} - \bar{x}_{yj})^2}{n_y - 1}$$

For the constant covariance matrix case, the pooled estimate of the covariance matrix was used.

The k-nearest neighborhood classifier was based on correlation between two mRNA samples. That is, with gene expression data $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{ip})$ and $\boldsymbol{x}_{i'} = (x_{i'1}, \cdots, x_{i'p})$ for two cases, the similarity measure was taken to be:

$$r_{ii'} = \frac{\sum_{j=1}^{p} (x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_{j=1}^{p} (x_{ij} - \bar{x}_i)^2 \sum_{j=1}^{p} (x_{i'j} - \bar{x}_{i'})^2}}$$

To classify an observation $\boldsymbol{x}$ in the test set, the rule first finds the k-nearest observations to $\boldsymbol{x}$ in the learning set, and then assigns $\boldsymbol{x}$ to the most frequent class. The number k was chosen by cross validation.

Fisher linear discriminant analysis and classification trees were also included in the comparison.

In the main comparison, for each learning set/test set run, the d genes with the largest BSS/WSS were selected using the learning set. Then, the rules are developed on the learning set using the selected d genes, and the error rate was estimated on the test set. This entire procedure was repeated 150 times.

Dudoit et al. (2000) also considered recent machine learning approaches such as bagging and boosting in the comparison. In the simplest form of bagging, perturbed learning sets of the same size as the original learning set are formed by sampling at random with replacement from the learning set. Predictors are then built for each perturbed dataset and aggregated by plurality voting; that is, the estimated class of $\boldsymbol{x}$ is the class having the plurality vote.

In boosting, the data are re-sampled adaptively using a weighted sampling scheme. The weights are determined adaptively with more weights allocated to the cases that are more often misclassified. The aggregation of predictors is performed by weighted voting, with earlier samples given less weight.

Note that bagging is a special case of boosting, where the re-sampling probabilities are uniform at each step and the perturbed predictors are given equal weight in the voting.

3.3 Molecular Classification of Cancer

Golub et al. (1999) describes an approach to cancer classification based on gene expression monitoring by DNA microarray, with application to data on human acute leukemia.

Gene selection steps to reduce the dimensionality of the data are employed prior to classification. Genes are chosen that display the best separation between means for the two classes, as measured by the "correlation" $p(j,c)$

$$p(j,c) = \frac{\mu_1(j)-\mu_2(j)}{\sigma_1(j)+\sigma_2(j)} \tag{3.4}$$

where $\mu_1(j),\sigma_1(j)$ and $\mu_2(j),\sigma_2(j)$ are the mean and standard deviation for values of gene j among the training samples of class 1 and 2 respectively.

Genes with largest value of $|p(g,c)|$ are selected. Given a set of d informative genes and a test sample vector $\boldsymbol{x} = (x_1,\cdots,x_d)$, the vote of gene j is given by:

$$v_j = p(j,c)(x_j - b_j)$$

where

$$b_j = \frac{\mu_1(j)+\mu_2(j)}{2}\text{, and } p(j,c) \text{ is given by (3.4).}$$

A positive value of v_j indicates a vote for class 1, while a negative value of v_j indicates a vote for class 2. Then the total vote V_1 for class 1 and the total vote V_2 for class 2 are obtained as follows:

$$V_1 = \sum_j \max(0, v_j) \quad \text{and} \quad V_2 = \sum_j \max(0, -v_j)$$

The votes are summed to determine the winning class. The vote for the winning class is $V_{win} = max(V_1, V_2)$.

For each prediction made by the classifier, Golub et al. (1999) also defined a prediction strength PS:

$$\text{PS} = \frac{V_{win} - V_{loss}}{V_{win} + V_{loss}}\text{, where} \quad V_{loss} = min(V_1, V_2)$$

This reflects the relative margin of victory of the votes. The sample was assigned to the winning class if PS exceeded a predetermined threshold of .3 and was otherwise considered uncertain.

This methodology, which is only applicable to datasets with two classes, was shown to be a variant of diagonal linear discriminant analysis (Dudoit et al., 2000). Note also that $\sigma_1(j)+\sigma_2(j)$ is an unusual way of calculating the standard error of a difference.

In addition to class prediction, the authors used self-organizing maps (SOM) for class discovery, and the cancer groups revealed by this method were compared to already known cancer types. This procedure automatically discovered the distinction between acute lymphoblastic leukemia and acute myeloid leukemia without prior knowledge of these classes.

CHAPTER 4

DATA SETS AND PREPROCESSING

4.1 Data Sets

Generally, our datasets have gene expression data x_{ij} for genes (features) $j=1,2,\cdots,p$ and cases (tissue sample) $i=1,2,\cdots,n$. The cancer type for case i is denoted by y_i. The expression data x_{ij} might be from a cDNA microarray, in which case it represents the log red-to-green ratios of a target sample (red) relative to a reference sample (green). Alternatively x_{ij} might be the expression level from an oligonucleotide array. Table 1 gives a summary of the data sets used in this paper.

4.1.1 NCI60 Data

We obtained this from http://genome-www.stanford.edu/nci60 (Ross et al., 2000). The data set comprises a set of 9,703 gene expression profiles among the 60 cell lines that are used by the Development Therapeutic program (DTP) of the National Cancer Institute (NCI) to screen potential anticancer drugs. The NICI60 set includes cell lines that are derived from cancers of colon, renal, ovarian, breast, prostate, non-small-cell-lung-carcinoma (NSCLC), central nervous system (CNS), leukemia and melanoma origins, as well as one unknown.

The expression level considered here is relative to the expression levels of a suitably defined common reference sample. That is, each of the hybridizations compares Cy5-labeled cDNA reverse transcribed from mRNA of one of the test cells with Cy3-labelled cDNA reverse transcribed from mRNA of the reference sample. The reference sample was prepared by combining an equal mixture of mRNA from 12 of the cell lines.

Table 1: Data set summary

Dataset	# of genes	Classes	# of cases	Type
NCI60	7661	Colon	7	Log red-to-green ratio
		Renal	9	
		Ovarian	6	
		Breast	9	
		NSCLC	9	
		CNS	5	
		Leukemia	8	
		Melanoma	8	
			Total = 61	
Lymphoma	6475	DLBCL	59	Log red-to-green ratio
		FLL	9	
		CLL	12	
			Total = 80	
Leukemia	7129	ALL: B-cell	38	Oligonucleotide
		T-cell	9	
		AML	25	
			Total = 72	

To assess the contribution of artefact sources of variation in the measured expression patterns, breast and leukemia cell lines were each grown in three independent cultures, and the entire process was carried out independently on mRNA extracted from each culture.

Because of their small class sizes, we have excluded the two prostate cell lines from the study, as well as the unknown cell line observation. Also genes with more than two missing data points are excluded from the study.

4.1.2 Lymphoma Data

The data used in this study were obtained from Alizadeh et al., (2000): http://llmpp.nih.gov/lymphoma. To characterize gene expression patterns in the three most prevalent adult lymphoid malignancies, Diffuse Large B-cell lymphoma (DLBCL), follicular lymphoma (FL) and chronic lymphocytic leukemia (CLL), Alizadeh et al. (2000) designed a specialized microarray called the 'Lymphochip' selecting genes that are preferentially expressed in lymphoid cells and genes with known or suspected roles in processes important in immunology or cancer.

Fluorescent cDNA probes labeled with the Cy5 dye were prepared from each experimental mRNA sample. A reference cDNA probe, labeled with Cy3 dye, was prepared from a pool of mRNAs isolated from nine different lymphoma cell lines. Each Cy5 labeled cDNA probe was combined with the Cy3 labeled reference probe and the mixture was hybridized to the microarray.

Of the total measurements 6.81% had missing values due to insufficient resolution or image corruption. After screening out genes with more than two missing data points and mRNA samples that are unreliable, the data were summarized into an 80 x 6475 matrix.

4.1.3 Leukemia Data

This data set was obtained from http://www-genome.wi.mit.edu/MPR/ (Golub et al., 1999). It contains gene expression measurements corresponding to 47 acute leukemia lymphoblastic leukemia (ALL; 38 B-cell and 9 T-cell) and 25 acute myeloid leukemia (AML) samples from bone marrow and peripheral blood.

In the article they used a learning set of 38 mRNA (27 ALL and 11 AML) bone marrow samples obtained from acute leukemia patients at the time of diagnosis and a test set of 34 (20 ALL and 14 AML) mRNA samples obtained from bone marrow and peripheral blood cells. The observations in the two data sets came from different labs and were collected at different times.

In each chip, the mRNA prepared from the cells was hybridized to the high-density oligonucleotide microarrays and then a quantitative expression level was measured for each gene. Intensity values have been rescaled so that overall intensities for each chip are equivalent. This rescaling was done by fitting a linear regression model using the intensities of all genes in the first case (baseline) and each of the other cases. 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 (case) in the dataset except the baseline, which received a rescaling factor of one.

4.2 Preprocessing

It is common practice to use the correlation between the gene expression profiles of two mRNA samples to measure their similarity. Consequently, we standardized observations to have mean zero and variance one across variables (genes). I.e., the observations in each of the n rows in the data matrix have mean zero and variance one, so the correlation between two rows is proportional to the inner product.

4.2.1 Missing Data Imputation

Some of the arrays in the NCI60 and lymphoma dataset contain a number of genes with unreliable or missing data. Unfortunately, all of the statistical methods considered here require a complete dataset.

All the genes with more than two missing data points are excluded from the study. However we used the k-nearest neighbor imputation method, introduced by Troyanskaya, Cantor, Sherlock, Brown, Hastie, Tibshirani, Botstein and Altman (2001), to estimate the missing data points of genes with one or two missing data points. This method selects k genes with expression profiles similar to the gene with the missing data point based on some metric for gene similarity e.g., Pearson correlation coefficient or Euclidian distance. If the genes are standardized to have a zero mean and unit variance it can easily be shown that the Euclidian distance and correlation are equivalent.

For each gene with missing data points, the algorithm would first find the k-nearest genes that have an expression profile for each of the missing entries. The algorithm then uses the average of the corresponding entries of the k nearest genes to estimate the missing value. The method is relatively insensitive to the choice of k within the range of 10 to 20 neighbors. We used k = 10.

4.2.1 Dimension Reduction

The high dimensionality of the expression data presents a serious challenge to statistical pattern recognition methods. The number of genes varies from 5,000 to 20,000, while the number of cases in each class varies from 5 to 60.

Given a large set of potential features (genes). dimension reduction can be achieved in several ways. One approach is to identify variables that can be eliminated from the classification problem. Thus, the task is to select d features out of the available p measurements. Examples are forward and backward selection. With large number of potential features however, such methods are likely to be ineffective.

A second approach is to find a transformation from the p dimensional space to some lower-dimensional space. Thus, the aim is to replace the original variables by a smaller set of derived variables. The transformation can be a linear or nonlinear combination of the original variables, and may be supervised or unsupervised. In the supervised case, the transformation makes use of the class label information.

Both of these approaches require the optimization of some criterion function $\boldsymbol{D}$. An example is the probability of misclassification. Obtaining a minimum probability of misclassification is often the aim in classification problem.

For feature selection, the optimization is over the set χ_d of all possible subsets of size d of the p possible measurements $x_1, x_2, \cdots, x_p$. Thus, we look for a subset $\tilde{\chi}$ for which $D(\tilde{\chi}) = \min_{\chi \in \chi_d} D(\chi)$

In feature extraction, the optimization is performed over a family of transformations of the variables. The family is usually specified (for example a linear transformation of the variables) and we seek the transformation $\tilde{A}$ for which

$$\tilde{A} = \arg\min\{ D(A) : A \in \mathcal{A}\},$$

where $\mathcal{A}$ is the set of allowable transformations. The feature vector is then $\boldsymbol{x}^* = \tilde{A}(\boldsymbol{x})$

To reduce the dimensionality of the datasets considered in this paper, we applied the k-means clustering criterion to the genes. Let $\boldsymbol{x}_j \in \Re^n$ represent the expression profile vector for gene j, for $j = 1, 2, \cdots, p$. The aim is to find the set of centroids (cluster centers) $\zeta_1, \zeta_2, \cdots, \zeta_g$ $(g < p)$ that minimizes

$$\sum_{j=1}^{p} \min_l \left\{ \left\| \boldsymbol{x}_j - \zeta_l \right\|^2, \ l = 1, 2, \cdots, g \right\} \qquad (4.1)$$

Then, $\zeta_1, \zeta_2, \cdots, \zeta_g$ will be used as features to develop the classifier.

This can be viewed as feature extraction. Define the transformation A as follows:

$$A_g(\mathbf{X}) : \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_p\} \rightarrow \{\zeta_1, \cdots, \zeta_g\}$$

where the subscript in A is used to indicate that the transformation is defined for fixed g and **X** represent the data matrix.

Then our aim is to find the value of g that optimizes the criterion $\boldsymbol{D}$. The possible values of g are from 1 to p. Optimizing $\boldsymbol{D}$ over these values however is highly computational. In this paper, we tried to select g over some arbitrarily specified values.

Each $\boldsymbol{x}_j$ has been standardized in such a way that $\boldsymbol{x}_j^T \boldsymbol{1}_n = 0$ and $\boldsymbol{x}_j^T \boldsymbol{x}_j = 1$. Thus, minimizing (4.1) is equivalent to clustering of genes based on their correlation; that is maximizing

$$\sum_{j=1}^{p} \max_{l} \left\{ \boldsymbol{x}_j^T \zeta_l \, , \, l = 1, 2, \cdots, g \right\}$$

assuming $\|\zeta_l\|^2 = 1$.

The optimization can be carried out by applying vector quantization (Gersho and Gray 1992), a stochastic gradient algorithm that repeatedly samples $\boldsymbol{x}_j$ and then updates the nearest ζ_l to $\boldsymbol{x}_j$.

Hooper (1999, 2001) employed vector quantization to initialize reference points for his reference point logistic (RPL) classification methods. We modified his vector quantization subroutine to calculate cluster centroids for the genes.

CHAPTER 5

DISCUSSION OF STATISTICAL METHODS USED

The framework for the pattern recognition problem in its simplest form is as follows: We know an item belongs to a set $\{1,2,\cdots,J\}$ of the possible classes. We do not know the particular class; however, we do have a measurement vector (features) associated with the item. Let Y denote the unknown class and let X represent a vector of p features measured. We assume we have a set of features for a number of similar items with known class $\{(\boldsymbol{x}_i, y_i), i=1,2,\cdots,n\}$ (the training set) that we use to design the classifier. We may distinguish three kinds of analysis:

- Discriminant Analysis: Estimate the conditional distribution of X given Y and uses the result to describe the nature and extent of differences among classes.
- Regression: Estimate the conditional distribution of Y given X.
- Classification: Predict Y given X.

The three analyses are related, with regression often following from discriminate analysis, and classification from regression. Given prior probabilities $p(y)$ and conditional densities $f(\boldsymbol{x} \mid y)$, one can find $p(y \mid \boldsymbol{x})$ using Bayes's Theorem:

$$p(y \mid \boldsymbol{x}) = \frac{p(y) f(\boldsymbol{x} \mid y)}{f(\boldsymbol{x})}$$

where

$$f(\boldsymbol{x}) = \sum_{y'=1}^{J} p(y') f(\boldsymbol{x} \mid y')$$

Let loss function $L(y', y)$ be the loss incurred when class y' is predicted and y is in fact the true class. The optimal classification rule predicts the class y' minimizing the conditional expected loss: $\sum_{y=1}^{J} L(y', y)\, p(y \mid x)$.

Under simple misclassification loss: $L(y', y) = \begin{cases} 1 & \text{when } y' \neq y \\ 0 & \text{when } y' = y \end{cases}$ another way to write the optimal rule is to assign $\boldsymbol{x}$ to class y' that maximizes $p(y' \mid \boldsymbol{x})$ or equivalently that maximizes $p(y')f(\boldsymbol{x} \mid y')$. This follows from Bayes' Theorem, and since $f(\boldsymbol{x})$ is independent of the class.

5.1 Reference Point Logistic (RPL) Regression

Reference point logistic (RPL) regression, introduced by Hooper (2001), is a generalization of logistic regression model that retains its simplicity when appropriate but allows greater flexibility when needed. It is closely related to a method developed by Hooper (1999) for constructing classification rules. Both are based on the same parametric family of functions and the same optimization technique but differ in choice of loss function and interpretation of the inferential methods. I.e., RPL regression produces an estimate of $p(y \mid x)$, which can be used to define a classification rule. RPL classification (Hooper 1999) produces only a classification rule.

An RPL regression model expresses $p(y \mid x)$ in terms of proximity between x and the reference points for each class. Given the total number of reference points

K and the number K_y of reference points assigned to class y, the vectors $\xi_k \in \Re^p$, scalars $\gamma_k \in \Re$, and a positive scale parameter τ, for $k = 1, 2, \cdots, K$, parameterize the model.

Define the class assignment function as: $cls(k) = y \quad if \sum_{m<y} K_m < k \le \sum_{m \le y} K_m$ and call ξ_k a reference point for $cls(k)$.

First define the RPL basis function $w_k(\boldsymbol{x})$ as a normalized exponential function of squared distance between the observed feature vector x and the reference points in the feature space:

$$w_k(\boldsymbol{x}) = \frac{\exp(\gamma_k - \tau^{-2} \| \boldsymbol{x} - \xi_k \|^2)}{\sum_m \exp(\gamma_m - \tau^{-2} \| \boldsymbol{x} - \xi_m \|^2)} \tag{5.1}$$

Then, the RPL model assumes

$$p(y \mid x) = \sum_{k=1}^{K} I\{cls(k) = y\}\, w_k(x).$$

Some algebraic manipulation of (5.1) gives

$$w_k(\boldsymbol{x}) = \frac{\exp\ (\alpha_k + \beta_k^T \boldsymbol{x})}{\sum_m \exp\ (\alpha_m + \beta_m^T \boldsymbol{x})}$$

where $\qquad \alpha_k = \gamma_k - \tau^{-2} \|\xi_k\|^2 - (\gamma_m - \tau^{-2} \|\xi_m\|^2)$

$\qquad\qquad \beta_k = 2\tau^{-2}(\xi_k - \xi_K)$

Note that $\alpha_K = 0$ and $\beta_K = 0$. This parameterization shows that $w_k(.)$ are functions used in multiple logistic models. Moreover, the RPL regression model is equivalent to a logistic regression model if one reference point is assigned to each class.

In the parameterization (5.1), the scale parameter τ and one pair (γ_k, ξ_k) are redundant; for example setting $\tau=1$, $\xi_K=0$ and $\gamma_K=0$, one can obtain arbitrary values of α_k and β_k for all $k \neq K$. When fitting the RPL model, redundant parameters are not removed because an over-parameterized model facilitates selection of initial parameters when maximizing the likelihood. The goal is to obtain a good estimator for $p(y \mid x)$. To achieve this goal, it is not necessary that parameter be identifiable or interpretable.

Given K and $\{K_y, y=1,\cdots,J\}$, the goal is to find parameter estimates that minimize the risk $\mathrm{E}_P[-\log p(Y \mid X)]$, where P is the joint distribution of (X,Y). To this end we attempt to minimize a training risk $\mathrm{E}_{\hat{P}}[-\log p(Y \mid X)]$, where $\mathrm{E}_{\hat{P}}$ is the expected value when sampling from an estimate $\hat{P}$ of the joint distribution of (X,Y).

An estimate $\hat{P}$ is based on a training set $\{(x_i, y_i), i=1,2,\cdots,n\}$. In the simplest case, $\hat{P}$ assigns probability $1/n$ to each case in the training set. In Hooper (1999, 2001) a more general class of estimators $\hat{P}$, that allows flexible choice of prior probabilities and the option of smoothing to obtain a density estimate of $f(x \mid y)$, are used. The author defined $\hat{P}$ in terms of how the pairs (X,Y) can be sampled from $\hat{P}$. Let $\hat{p}(y)$ be an estimate of the prior probability of class y. If the training set is a simple random sample, then $\hat{p}(y)$ is usually the sample proportion, n_y/n in class y. In some applications, $\hat{p}(y)$ may be estimated using additional information or may be specified arbitrarily.

Let U be a random variable taking values in $\{1,2,\cdots,n\}$ such that (i) $Pr\{y_U = j\} = \hat{p}(j)$ and (ii) the conditional distribution of U given $y_u = j$ is uniform on $\{i : y_i = j\}$. Let Z be a p-dimensional vector of independent standard normal random variables, with Z and U independent. The estimate $\hat{P}$ is defined as the distribution of $(x_U + \lambda Z, y_u)$, where $\lambda \geq 0$. In our work, we set $\lambda = 0$. This "smoothing" or "jittering" on the $\boldsymbol{x}$-vectors is usually not helpful when the conditional log likelihood criterion is employed.

The training algorithm is as follows. Minimization begins by specifying initial parameter values. The algorithm uses the k-means clustering criterion to choose initial reference points ξ_k^0; that is, for class j, it finds a set of K_j points ξ_k^0 to minimize

$$\sum_{\{i: y_i = j\}} min\{ \left\| x_i - \xi_k^0 \right\|^2 : cls(k) = j \}.$$

Set $\gamma_k^0 = 0$. Set $\tau = c_\tau s_{nn}$, where c_τ = 1.00 and s_{nn} is the average distance between nearest neighbors among the K initial reference points ξ_k^0.

After specifying the initial parameter values, the training risk is minimized by stochastic approximation where minimization is carried out over $\{\gamma_k, \xi_k\}$ with τ fixed. In each of the iterations, the algorithm samples an observation $(x, y) = (x_u + \lambda Z, y_u)$ from $\hat{P}$ and updates the parameter estimates.

Differentiation of log $p(y \mid x)$ with respect to γ_k and ξ_k gives the following updating formulae at the *mth* iteration:

$$\gamma_k \leftarrow \gamma_k + a_m^{\gamma} h_k(x,y)$$

$$\xi_k \leftarrow \xi_k + a_m^{\xi} h_k(x,y)(x - \xi_k)$$

where

$$h_k(x,y) = \left[\frac{I\{cls(k) = y\}}{p(y \mid x)} - 1 \right] w_k(x)$$

The number of reference points per class K_y and the smoothing parameter λ can be selected empirically using cross-validated risk estimates. When applying RPL to the microarray data, we found that choosing a single reference point per class and setting $\lambda = 0$ gave a better performance.

The RPL regression model possesses two invariance properties. First, RPL regression is invariant under affine transformation, provided the non-smoothed training risk; that is $\lambda = 0$, is used and one is able to minimize this training risk. The usefulness of this result is limited in the following sense. The effectiveness of stochastic approximation in minimizing the training risk depends to some extent on the initial reference points. However their selection is not equivariant under affine transformation of the feature vector. Consequently, a judicious transformation of the feature can improve estimation of the RPL model.

The second invariance property concerns specification of prior probabilities. Given a conditional density model $f(x \mid y)$, let $p^*(y \mid x)$ and $p^{\#}(y \mid x)$ denote the conditional probability models determined by priors $p^{*}(y)$ and $p^{\bullet}(y)$ respectively. Assuming all priors are positive, the two conditional models are related as follows:

$$p^{\bullet}(y|x)=\frac{p^{*}(y|x)p^{\bullet}(y)/p^{*}(y)}{\sum_{j=1}^{J}p^{*}(j|x)p^{\bullet}(j)/p^{*}(j)} \tag{5.2}$$

If $p^{*}(y|x)$ is an RPL model with γ_k^{*}, then $p^{\bullet}(y|x)$ is also an RPL model with $\gamma_k^{\bullet}=\gamma_k^{*}+\log\{p^{\bullet}(y)/p^{*}(y)\}$ for $y=cls(k)$. The other parameters ξ_k and τ are the same in both models. This invariance property suggests a re-weighting strategy when estimating an RPL model. Suppose $p^{\bullet}(y)$ represent realistic prior probabilities. One can assign arbitrary prior probabilities $\hat{p}(y)=p^{*}(y)$, apply the RPL training algorithm to estimate $p^{*}(y|x)$ and then apply (5.2) to estimate $p^{\bullet}(y|x)$.

The re-weighting strategy is helpful when priors are highly unbalanced. Small priors can create instability during training, with parameter estimates diverging and conditional probability associated with these small priors being underestimated. In this situation it is useful to use more balanced priors when fitting the RPL model, and then adjust the estimates obtained using (5.2). This strategy is used in lymphoma data set. Equal prior probabilities are assigned when fitting the RPL model and we applied the reweighing strategy using the empirical priors.

5.4 Classifiers Based on the Normal Distribution

This section will discuss the two most widely used classifiers based on the normal distribution: Linear Discriminant Analysis and Quadratic Discriminant Analysis.

5.2.1 Linear Discriminant Analysis (LDA)

LDA assumes $f(\boldsymbol{x} \mid y)$ is a multivariate normal, with the mean vector μ_y depending on the class y and the covariance matrix being the same for all classes; that is

$$f(\boldsymbol{x} \mid y) = (2\pi)^{-\frac{p}{2}} |\Sigma|^{-\frac{1}{2}} exp\{ -\frac{1}{2}(\boldsymbol{x} - \mu_y)^T \Sigma^{-1} (\boldsymbol{x} - \mu_y) \}$$

The Bayes rule assigns $\boldsymbol{x}$ to class y that maximizes $f(\boldsymbol{x} \mid y)\, p(y)$, or equivalently $\log\{f(\boldsymbol{x} \mid y)\; p(y)\} \propto 2\log p(y) - (\boldsymbol{x} - \mu_y)^T \Sigma^{-1} (\boldsymbol{x} - \mu_y)$. Note that $(\boldsymbol{x} - \mu_y)^T \Sigma^{-1} (\boldsymbol{x} - \mu_y)$ is the square of Mahalanobis distance from $\boldsymbol{x}$ to the mean of class j. Since the quadratic term $\boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x}$ is common to all the classes the rule can be written as: assign $\boldsymbol{x}$ to class y that maximizes $2\log p(y) + 2\mu_y^T \Sigma^{-1} \boldsymbol{x} - \mu_y^T \Sigma^{-1} \mu_y$.

If we further assume the classes have equal priors then x is classified as coming from the nearest class, in the sense of having the smallest Mahalanobis distance to its mean. If in addition Σ is proportional to the identity matrix then distance is Euclidean distance.

The parameters μ_y and Σ are estimated from the training set by the sample mean vectors and pooled covariance matrix, respectively as follows:

$$\hat{\mu}_y = \frac{1}{n_y}\sum_{i=1}^{n} I(y_i = y) \boldsymbol{x}_i$$

$$\hat{\Sigma} = \sum_{y=1}^{J} \frac{n_y}{n} \hat{\Sigma}_y$$

where

$$\hat{\Sigma}_y = \frac{1}{n_y} \sum_{i=1}^{n} I(y_i = y)(\boldsymbol{x}_i - \hat{\mu}_y)(\boldsymbol{x}_i - \hat{\mu}_y)^T$$

Often the bias corrected estimator of Σ_y with divisor $n_y - 1$ is preferred. This makes no difference to the linear rule unless the prior probabilities differ, in which case the effect is to change the constant terms to reduce slightly the influence of the data term relative to the prior (Ripley, 1996 p. 36). If the training set is a simple random sample, then $p(y)$ is usually estimated by the sample proportion n_y/n in class y. In some applications, $p(y)$ may be estimated using additional information or may be specified arbitrarily.

5.4 Quadratic Discriminant Analysis

If we assume $f(\boldsymbol{x} \mid y)$ to have a multivariate normal distribution with both the mean vector and covariance matrix depending on the class y, the optimal rule assigns $\boldsymbol{x}$ to the class y maximizing:

$$2 \log p(y) - \log |\Sigma_y| - (\boldsymbol{x} - \mu_y)^T \Sigma_y^{-1} (\boldsymbol{x} - \mu_y) \qquad (5.3)$$

This is referred as the quadratic discriminant rule.

The number of estimated parameters has increased from $JP + \frac{p(p+1)}{2}$ for LDA to $JP + \frac{JP(p+1)}{2}$ for QDA.

LDA is quite robust to departures from the equal covariance assumptions (O'Neil, 1992), and may give better performance than the quadratic rule for normally distributed classes when Σ_y is unknown and the sample sizes are small. However, it is better to use the quadratic rule if the sample size is sufficient. Lachenbruch et al. (1973) investigated the robustness of the linear and quadratic rules to certain types of non-normality. LDA can be greatly affected by non-normality and, if possible, variables should be transformed to approximate normality before applying the rule.

Problems will occur in the quadratic rule classifier if any of $\hat{\Sigma}_y$ is singular. There are several alternatives commonly employed. One is simply to use a diagonal covariance matrices; that is $\hat{\Sigma}_y = diag(\hat{\sigma}_{y1}^2, \hat{\sigma}_{y2}^2, \cdots, \hat{\sigma}_{yp}^2)$. Under this assumption the quadratic rule (5.3) assigns $\boldsymbol{x}$ to class y that maximizes

$$2\log p(y) - \sum_{j=1}^{p}\left\{\frac{(x_j-\hat{\mu}_{yj})^2}{\hat{\sigma}_{yj}^2}+\log\hat{\sigma}_{yj}^2\right\}.$$

The corresponding linear rule under the assumption $\hat{\Sigma} = diag(\sigma_1^2,\cdots,\sigma_p^2)$ is to assign $\boldsymbol{x}$ to class y that maximizes

$$2\log p(y) \;-\; \sum_{j=1}^{p}\frac{(x_j-\mu_{yj})^2}{\sigma_j^2}.$$

Another approach is to project the data to a space in which $\hat{\Sigma}_j$ is non-singular, for example using a principal components analysis, and then to use the Gaussian classifier in the reduced dimension space.

5.3 K-Nearest Neighbors

Here the class conditional density $f(\boldsymbol{x}\mid y)$ is estimated by a k-nearest neighbor method as follows. Suppose we have a training sample of size n of which n_y observations are from class y and the hypersphere around $\boldsymbol{x}$ containing k nearest observations has volume $V(\boldsymbol{x})$ and contains $k_1,k_2,\cdots,k_J$ observations of class $1,2,\cdots,J$ respectively. Then the k-nearest neighbor estimate of $f(\boldsymbol{x}\mid y)$ is

$$\hat{f}(\boldsymbol{x}\mid y) = \frac{k_y}{n_y V(\boldsymbol{x})} \tag{5.4}$$

One thing to note about this density estimate is that it is not in fact a density. The integral under the curve is not one.

In k-nearest neighbor rules prior probabilities are estimated by sample proportions n_y/n. Then using this estimated prior and the density estimate in (5.4) leads immediately to the k-nearest neighbor rule: assign $\boldsymbol{x}$ to class y that maximizes

$\hat{f}(\boldsymbol{x} \mid y)\,\hat{p}(y) = \dfrac{k_y}{n_y V(\boldsymbol{x})}\dfrac{n_y}{n}$, or equivalently assign $\boldsymbol{x}$ to class y if $k_y \geq k_{y'}$ for all y'.

The k-nearest neighbor of a given observation vector $\boldsymbol{x}$ from among the training sets involves choice of a suitable metric. In some data sets, where the measurements are measured on different scales, some standardization is required.

The above rule assumes $p(y)$ is estimated by n_y/n; however it could be the case that our sample did not estimate priors correctly. The number of neighbor k is often chosen by cross validation.

5.4 Error Rate Estimation

The performance of a classifier is most often evaluated by its probability of misclassification (pmc). This probability can be estimated by the apparent error rate: the proportion of errors made when classifying training or a test data. If the training set is used the pmc will usually be biased downward because the data have been used twice, both to develop the rule and to evaluate its performance.

One way of avoiding this bias is to use a test set independent of the training set. A proportion of the data set is selected at random (usually about 10-30%) and used as test data. We train the classifier on the remaining data and then the error rate is estimated on the test data. The error rates are unbiased but can be highly variable.

One way of decreasing this variability is to repeatedly divide the data into test and training sets a large number of times and average of the error rates. Having to use a test set is often regarded as waste of data, which could otherwise have been used for training, but with large data sets this is not a major problem.

A method that is most suitable for intermediate sample sizes is cross-validation. We first divide the data randomly into v groups so that their sizes are as nearly equal as possible. Each time, one of the v subsets is used as the test set and the other $v-1$ subsets are put together to form a training set. Then the average error across all v trials is computed. Every data point gets to be in a test set exactly once, and gets to be in a training set $v-1$ times.

In our investigation we used repeated 3-fold cross-validation, a combination of 3-fold cross validation and the repeated learning-testing method; that is, we repeated the method of 3-fold cross validation B times. On the bth repetition we randomly split the data in 3 groups and get its cross-validated error rate $p\hat{m}c_b$ as described above. Finally we calculated the mean and median of these B estimates. We also obtained the standard deviation and range of these estimates.

We can also estimate the error rate conditioning for each class, just by counting within each class. These conditional error estimates $p\hat{m}c_y$ can be combined with prior probabilities $p(j)$ to obtain an alternative estimate:

$$p\hat{m}c = \sum_{y=1}^{J} p(\ y\ )\, p\hat{m}c(\ y\ )$$

This estimator is also unbiased, assuming the prior probabilities are known or estimated in the usual unbiased way, but it is undefined if $n_y = 0$ for any class.

5.4 Study Design

We first arbitrarily select a set of $g = 10, 20, 30, 40, 50, 100, 150, \text{and } 200$ gene centroid vectors using the vector quantization algorithm discussed in Section 4.2.2.

Let $p\hat{m}c(g)$ denote the v-fold cross-validated error rate obtained by using g centroid vectors as a feature to develop the classifier. Then, for each classifier, we select g that minimizes $p\hat{m}c(g)$, i.e.

$$g^{*} = \arg\min p\hat{m}c(g)$$

Then, for this selected g^{*} the following procedure was repeated 150 times: at the bth iteration the data was randomly divided into three groups. We train the models (RPL, LDA, QDA and k-nn) three times, each time leaving out one of the subsets from the training, but using the omitted subset to compute the error rate. Denote the average of these three estimates by $p\hat{m}c_b$. For a more effective comparison, the same partitioning of data was used for all classification methods.

This procedure was carried out for all three of the data sets described in Section 4.1.

In RPL regression, we used one reference point for class, and we set λ to be zero.

The number of neighbor k in k-nn is chosen by 3-fold cross-validation. This is done for a number of ks and the k for which the error rate is smallest is retained for later use.

Note that k-nn rule uses an empirical prior probability in an implicit way. Thus, to make our comparison more effective, we adopted the empirical prior probabilities for the other classifiers.

CHAPTER 6

DISCUSSION OF RESULTS

The number of centroid vectors g to be used in the final classifier is selected by cross validation. Figures 3, 4, 6 and 7 displays the plot of cross-validated risk versus g for each of the classifiers considered here. In the k-nearest neighbor classifier, to choose the g, we used $k=1$.

RPL Regression: Setting $g=50$ for lymphoma data, $g=40$ for leukemia two classes problem, and $g = 20$ for leukemia three classes problem seems to give a better performance. See Figure 3. For the NCI60 data set $g=100$ seems to give a slightly lower error rate. The summary error rates for these values of g are listed in Table 2.

Table 2: Cross-validated error rate summary for RPL regression, B=150. We used one reference point per class and $\lambda=0$.

Data Set	g	Mean	Median	Standard Deviation	Range
Lymphoma	50	0.0855	0.0822	0.0305	0.1448
Leukemia: Two class	40	0.0460	0.0417	0.0224	0.1111
Three class	20	0.1187	0.1111	0.0342	0.1944
NCI60	100	0.3082	0.3090	0.0515	0.2693

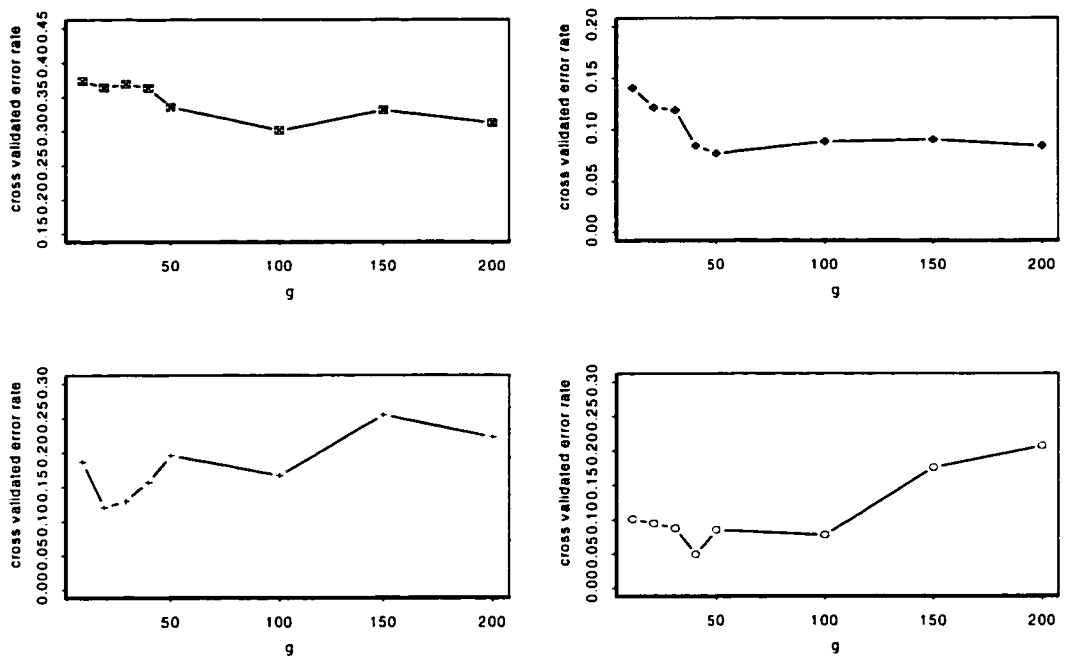

Figure 3: RPL regression. Plot of cross-validated error rate for different values of g. The top left is for NCI60 data set, and the top right is for lymphoma data set. The bottom left displays leukemia data with three classes, and the bottom right is for leukemia data with two classes.

K-nearest Neighbor: Table 3 lists the selected g for each data set together with its summary error rate for $B=150$ runs. For all the data sets, $g=30$ seems to give a lower error rates (see Figure 4). The parameter k of the nearest neighbor classifier is selected by cross validation and is usually one. This suggests very good predictors can be obtained from the class of the case most highly correlated to the case to be predicted. Figure 5 displays the average error rate (over 20 runs) of each data set for different values of k.

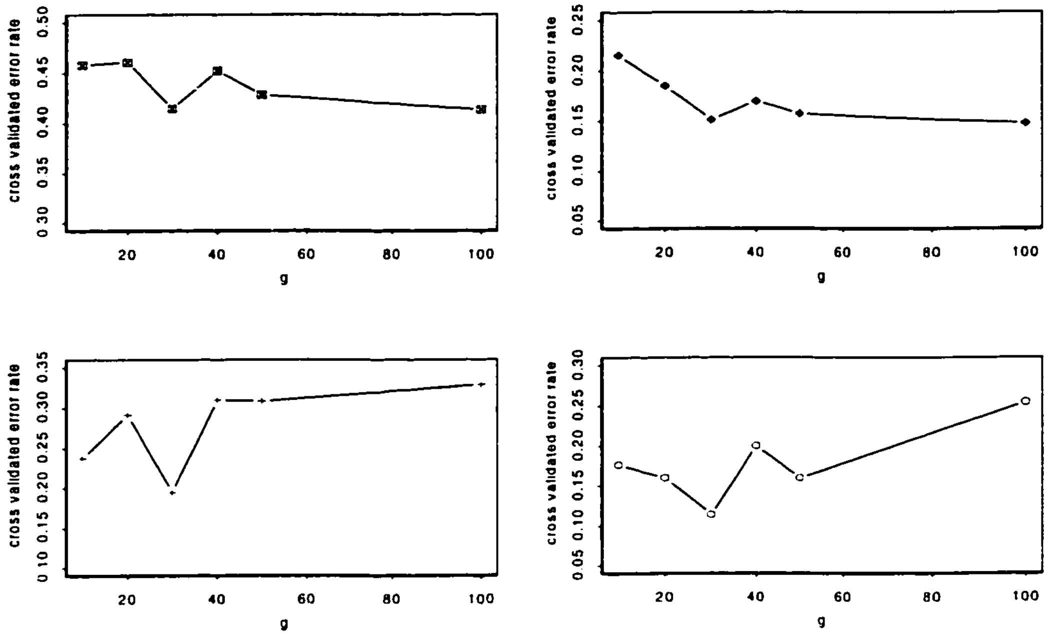

Figure 4: k-nearest neighbor rule. Plot of cross-validated error rate for different values of g. The top left is for NCI60 data set, and the top right is for lymphoma data set. The bottom left displays leukemia data with three classes, and the bottom right is for leukemia data with two classes.

Table 3: Cross-validated error rate summary for k-nn rule, B=150. We used k = 1 for all the data sets.

Data Set	*g*	Mean	Median	Standard Deviation	Range
Lymphoma	30	0.1476	0.1493	0.0303	0.1501
Leukemia: Two class	30	0.1195	0.1111	0.0244	0.1111
Three class	30	0.1897	0.1944	0.0308	0.1389
NCI60	30	0.4090	0.4087	0.0409	0.2151

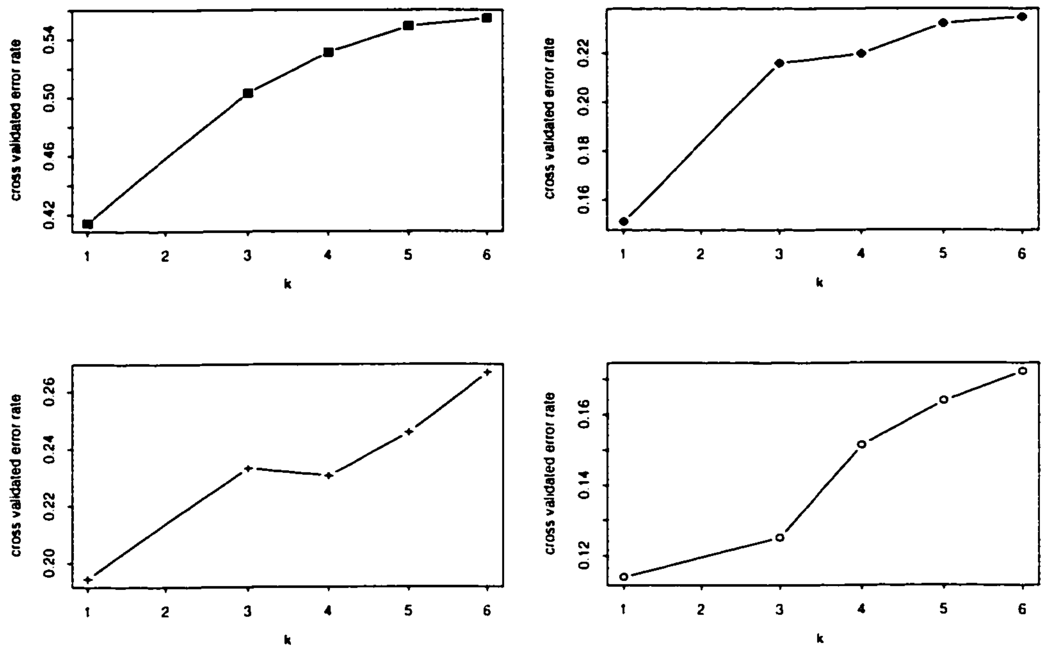

Figure 5: Cross-validated error rate of k-nearest neighbor for different values of k using $g=30$ for all data sets. The top left is for NCI60 data set, and the top right is for lymphoma data set. The bottom left displays leukemia data with three classes, and the bottom right is for leukemia data with two classes.

Quadratic Discriminant Analysis: The performance of DQDA seems insensitive to the choice of g. Setting g = 10 for the NCI60 data set and g = 30 for the other data sets gives a slightly smaller error rate as compared to other values. See Figure 6. The mean and median misclassification rates together with its standard deviation for these selected values of g are reported in Table 4. The performance of DQDA is worst in the NCI60 data set. This is probably due to the small number of observations in each class.

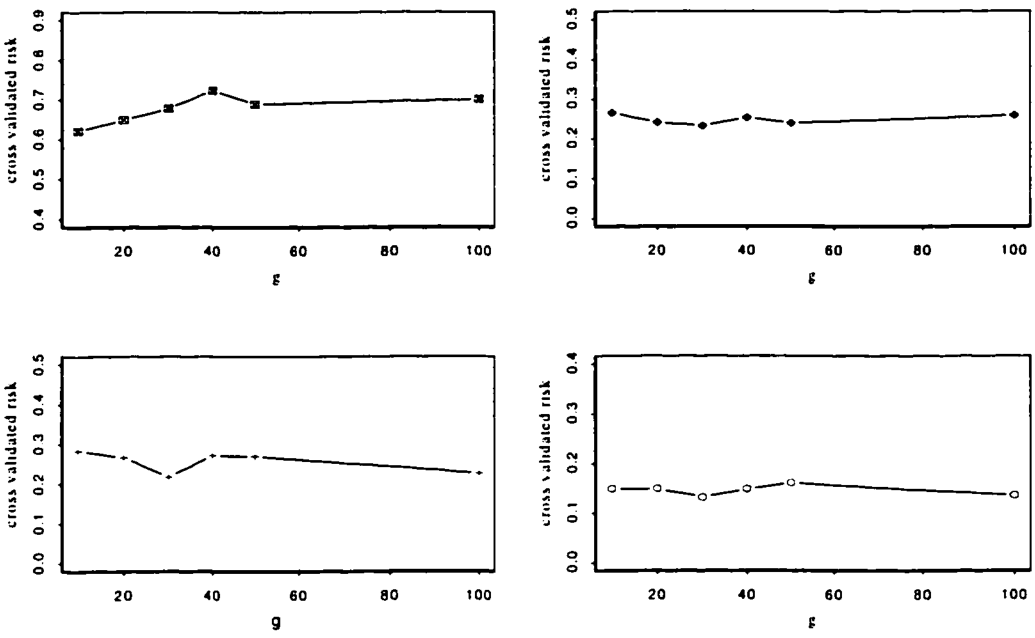

Figure 6: DQDA. Plot of cross-validated error rate for different values of g. The top left is for NCI60 data set, and the top right is for lymphoma data set. The bottom left displays leukemia data with three classes, and the bottom right is for leukemia data with two classes.

Table 4: Cross-validated error rate summary for DQDA, B = 150.

Data Set	*g*	Mean	Median	Standard Deviation	Range
Lymphoma	30	0.2249	0.2246	0.0234	0.2094
Leukemia: Two class	30	0.1306	0.1250	0.0249	0.1111
Three class	30	0.2235	0.2222	0.0322	0.1806
NCI60	10	0.6140	0.6095	0.0508	0.3103

Linear Discriminant Analysis: DLDA, which assumes a common diagonal covariance matrix, gives a better performance when g = 50 for the NCI60 data set, and when g = 30 for leukemia (both three classes and two classes problem) and lymphoma data set. See Figure 7. The summary misclassification rates together with its standard deviation for these selected values of g are listed in Table 5. With the exception of lymphoma data set, DLDA give lower misclassification rate than DQDA, which allow different diagonal covariance matrices. The performance of DLDA is specially striking for the NCI60 data set, where it performed better with pmc of approximately 36.25% (next to RPL regression with pmc ≈ 30.82%) than KNN (with pmc ≈ 40.90%) and DQDA (with pmc ≈ 61.40%).

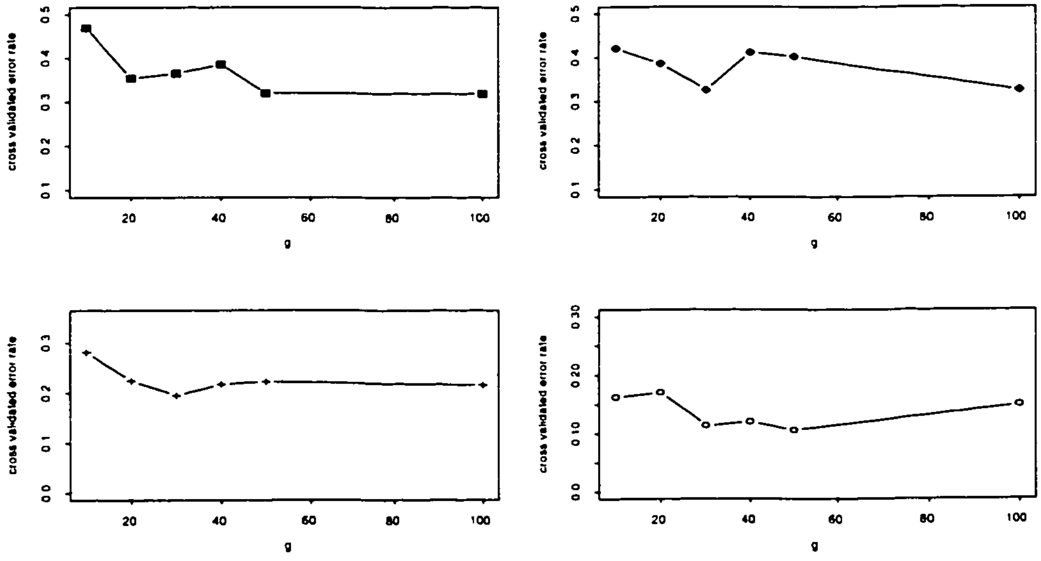

Figure 7: DLDA. Plot of cross-validated error rate for different values of g. The top left is for NCI60 data set, and the top right is for lymphoma data set. The bottom left displays leukemia data with three classes, and the bottom right is for leukemia data with two classes.

Table 5: Cross-validated error rate summary for DLDA, B = 150.

Data Set	g	Mean	Median	Standard Deviation	Range
Lymphoma	30	0.3287	0.3240	0.0429	0.2118
Leukemia: Two class	30	0.1155	0.1111	0.0283	0.1389
Three class	30	0.1839	0.1806	0.0306	0.1806
NCI60	50	0.3625	0.3611	0.0513	0.2786

To compare classifiers, we displayed the box plots of misclassification rates for each data set in Figures 8, 9, 10, and 11. RPL has a remarkably significant lower misclassification rate, for all the data sets, as compared to other classifiers. With the exception of lymphoma data set, the performance of k-nearest neighbor rule and DLDA, which assumes a common diagonal covariance matrix, has almost the same misclassification rates. DQDA, which allows different diagonal covariance matrices, have the highest misclassification rate for the leukemia and NCI60 data set. However, for the lymphoma data set, its performance is better than DLDA.

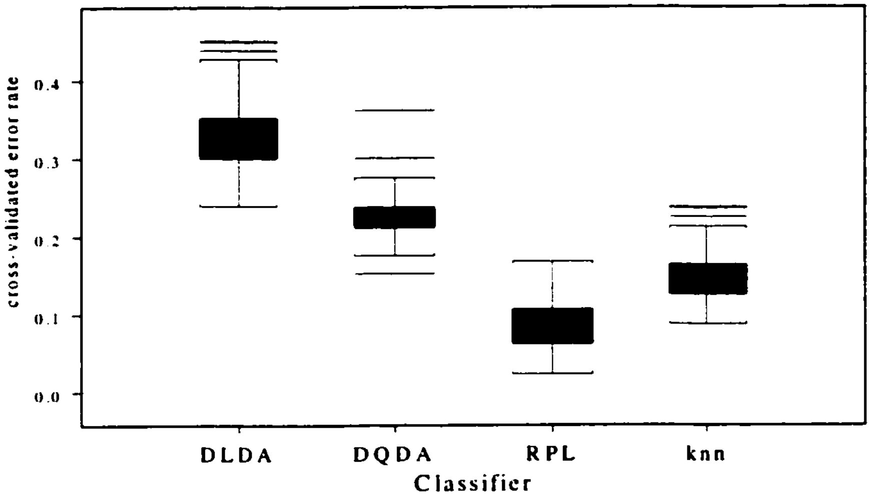

Figure 8: Box plots of 3-fold cross-validated misclassification rates for lymphoma data. We set g = 30 in DLDA, DQDA and k-nn. For RPL regression we set g = 50, B=150.

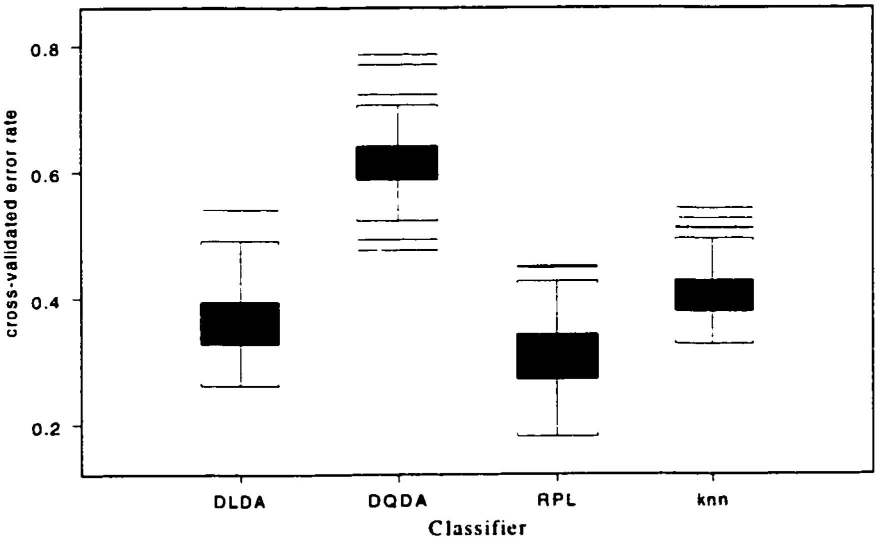

Figure 9: Box plots of 3-fold cross-validated misclassification rates for NCI60 data, B=150.

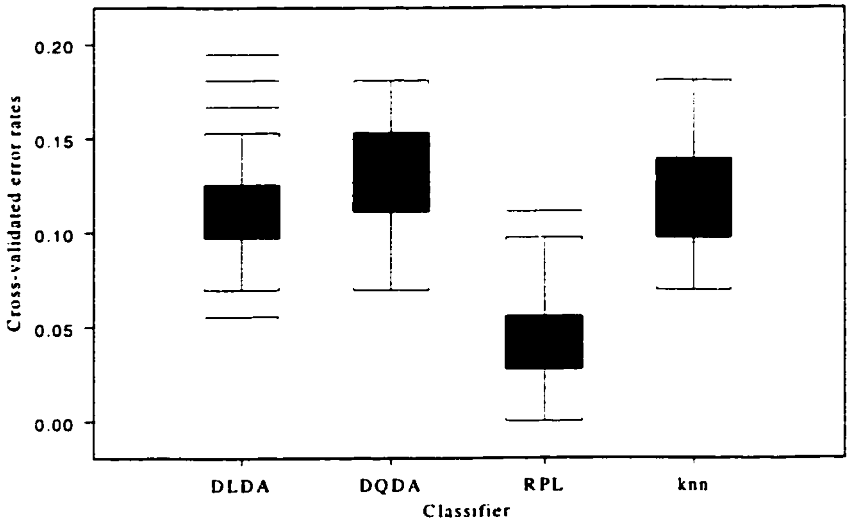

Figure 10: Box plots of 3-fold cross-validated misclassification rate for leukemia data, two classes, B = 150.

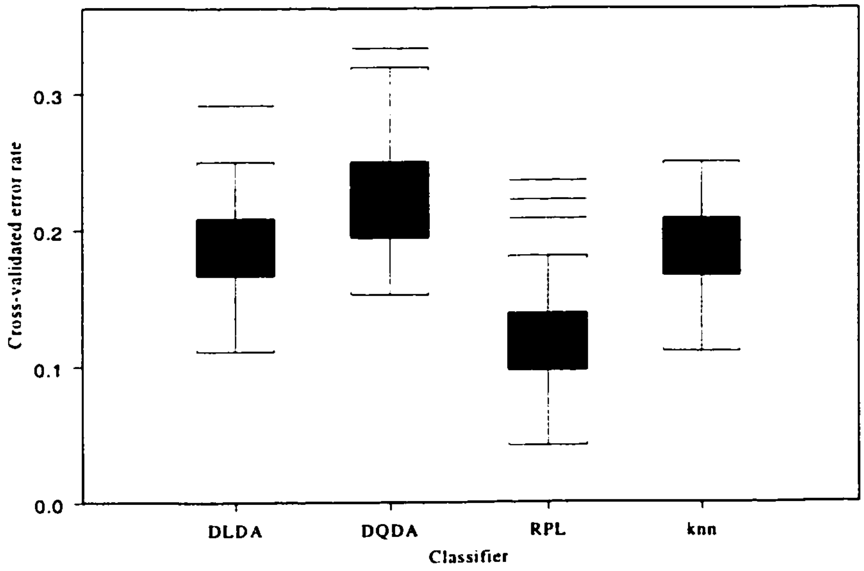

Figure 11: Box plots of 3-fold cross-validated misclassification rates for leukemia data, three classes.

For the lymphoma and leukemia data set, linear discriminant analysis with the unrestricted common covariance matrix has been included in the comparison. We could not run the "lda" procedure on the NCI60 data set. This is probably due small sample size. We must have $n - J \geq P$ for the pooled covariance matrix to be nonsingular.

For the lymphoma data set, it has impressively lower error rate as compared to the diagonal LDA, which ignores the correlations between genes. However, using the correlation between genes does not help that much to improve the performance in the leukemia two classes data set. The performance of LDA is slightly worse than DLDA in the leukemia data with three classes. Figure 12 displays the box plot misclassification rates of LDA and DLDA.

Dudoit et al. (2000) included DLDA, DQDA and k-nn in their comparison. The ranking of these classifiers in their paper is the almost the same as in ours. However, their reported error rates are lower than ours. This is probably due to the dimension reduction method employed. The authors selected p genes with the largest BSS/WSS ratios. This appears to select genes that provide better discrimination between classes.

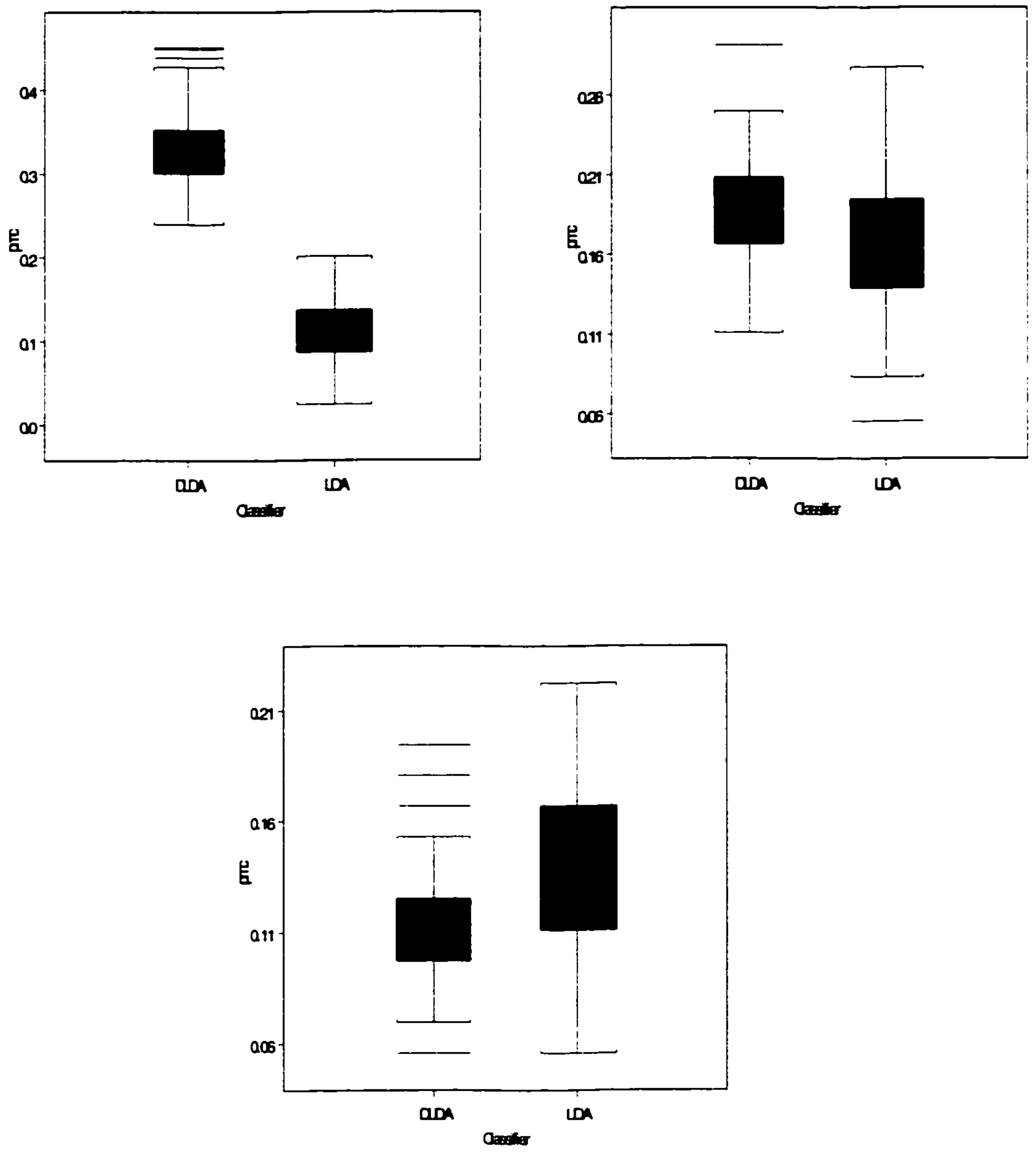

Figure 12: Box plot of misclassification error rates of LDA and DLDA. The top left is for lymphoma data set and the top right is for leukemia two classes data set. The bottom is for leukemia three classes problem.

CHAPTER 7

CONCLUSION

We have compared the performance of four different classifiers. With the exception of NCI60 data, the error rates seemed fairly low given the limited amount of data. The performance of classifiers on the NCI60 data set was much worse than on the other two data sets. This is probably due to the small class sizes. The ranking of classifiers, except in the lymphoma data where DQDA had a better performance than DLDA, were the same across data sets. RPL is the best, followed by k-nn and DLDA.

In Dudoit et al. (2000), DLDA was the best classifier for NCI60 data (with median error rate of ≈ 37%) and for leukemia data with two classes (with median error rate of ≈ 0%). For the lymphoma data and leukemia data with three classes, k-nearest neighbor rule performed better with median error rate of ≈ 0% for lymphoma data and ≈ 5%.

The approximate median error rates of RPL for the lymphoma, leukemia two classes, leukemia three classes, and NCI60 data sets are 8.22%, 4.17%, 11.11%, and 30.90% respectively. These error rates are higher than that of Dudoit et al. (2000). This is probably due to the dimension reduction technique we employed. RPL had out performed both DLDA and k-nearest neighbor rule that are developed using our dimension reduction technique.

Dudoit et al. (2000) estimated misclassification rates for the different classifiers based on repeated random divisions of each data set into a learning set and a test set comprising respectively two third and one third of the data (2:1 sampling scheme). The reason for using a 2:1 sampling scheme, rather than the standard 9:1 scheme, is the later scheme would result in a very small test sets and more difficult comparison among classifiers.

In this thesis, misclassification rates were estimated based on repeated 3-fold cross validation. Repeated 3-fold cross validation estimates are preferable than the repeated 2:1 learning testing estimates because the former has a smaller variance (Burman, 1989).

A very important issue that remains to be addressed is the choice of the number of centroids in the dimension reduction. We selected the g^* (from 10, 20, 30, 40, 50, 100 and 200) that minimized the misclassification rate, i.e. g^* was selected to match the classifier. Different starting values of g's may produce a different choice of g^*

References

Ash A. Alizadeh, Michael B. Eisen, R. Eric Davis, Chi Ma, Izidore S. Lossos, Andreas Rosenwald, Jennifer C. Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, John I. Powell, Liming Yang, Gerald E. Marti, Troy Moore, James Hudson Jr, Lisheng Lu, David B. Lewis, Robert Tibshirani, Gavin Sherlock, Wing C. Chan, Timothy C. Greiner, Dennis D. Wiesenberger, James O. Armitage, Roger Warnke, Ronald Levy, Wyndham Wilson, Michael R. Grever, John C. Byrd, David O. Brown and Louis M. Staudt (2000), Distinct Types of Diffuse Large B-cell Lymphoma identified by Gene Expression Profiling, *Nature*, 403, 503-511.

Prabir Burman (1989), A comparative study of ordinary cross-validation, v-fold cross-validation and repeated learning-testing methods, *Biometrika*, 76(3), 503-514.

Sandrine Dudoit, Jane Fridlyand, and Terence Speed (2000), Comparison of Discriminant Methods for Classification of Tumors using Gene Expression Data, *Technical report # 576*, Statistics Department, University of California, Berkeley.

A. Gersho and R.M. Gray (1992), *Vector Quantization and Signal Compression*, Dardrecht: Kluwer Academic Publisher.

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

Daniel L. Hartl (1991), *Basic Genetics*, Boston: Jones and Bartlett Publishers.

Trevor Hastie, Robert Tibshirani, David Botstein and Pat Brown (2000), Supervised Harvesting of Expression Trees, *Technical report,* Department of Statistics, Stanford University, Stanford.

Peter M. Hooper (1999), Reference Point Logistic Classification, *Journal of Classification,* 16, 91-116.

Peter M. Hooper, H. Zhang and David S. Wishart (2000), Prediction of Genetic Structure in Genomic DNA, using Reference Point Logistic Regression Models and Sequence Alignment, *Bioinformatics,* 16, 425-438.

Peter M. Hooper (2001), Reference Point Logistic Regression and the Identification of DNA Functional sites, *Journal of Classification,* 18, 81-107.

P. A. Lachenbruch, C. Sneeringer and L.T. Revo (1973), Robustness of the linear and quadratic discriminant functions to certain types non-normality, *Communications in Statistics,* 1(1) 39-56.

D. Michie, D.J. Spiegelhalter, C.C. Taylor (1994), *Machine Learning, Neural and Statistical Classification*, New York: Ellis Horwood.

T. J. O'Neill (1992), Error rates of non-Bayes classification rules and the robustness of Fisher linear discriminant function, *Biometrika*, 79(1), 177-184.

B. D Ripley (1996), *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University press.

Douglas T. Ross, Uwe Scherf, Michael B. Eisen, Charles M. Perou, Christian Rees, Paul Spellman, Vishwanath Iyer, Stefanie S. Jeffrey, Matt Van de Rijn, Mark Waltham, Alexander Pergamenschikov, Jeffery C.F. Lee, Deval Lashkari, Dari Shalon, Timothy G. Myers, John N. Weinstein, David Botstein and Patrick O. Brown (2000), Systematic Variation in Gene Expression Patterns in Human Cancer Cell Lines, *Nature Genetics*, 24, 227-235.

M. Schena, D. Shalon, R. Davis and P.O. Brown (1995), Quantitative Monitoring of Gene Expression Patterns with a cDNA Microarray, *Science*, 270, 467-470.

M. Schena (1999), *DNA Microarrays: A Practical Approach*, Oxford: Oxford University Press.

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein and Russ B. Altman (2001), Missing value estimation methods for DNA microarrays, *Bioinformatics*, 17(6), 520-525.

Appendix: Splus Code

```
#-------------------------------------------------------------------
#        Diagonal linear discriminant Analysis
#-------------------------------------------------------------------

library (section = MASS)  # for function max.col()
library(section = class) # for function k-nn

#-------------------------------------------------------------------
#             Diagonal Linear Discriminant Analysis
#             --------------------------------------

#  Diagonal LDA: assumes the class densities all have common
#                diagonal covariance matrix. The off-diagonal
#                elements of the covariance matrix are set to
#                zero.
#  Pre:
#  ====
#            Pi: prior probabilities provided by the user.
#    train.data: training data. The first column of the matrix
#                contains the class labels.
#     test.data: test data.
#             J: number of classes in training data.
#             p: number of features.
#  Post:
#  =====
#       Computes the mean vector and the diagonal covariance matrix
#       from the training data and returns the predicted class for
#       each case in the test data to the caller.
#-------------------------------------------------------------------
```

```
DLDA <- function(pi, train.data, test.data, J, p)
{
    mu <- matrix(0, nrow = J, ncol = p)      # for the mean vectors
    sigma <- matrix(0, nrow = p, ncol=p)     # for the cov matrix
    sigma.inv <- matrix(0, nrow = p, ncol=p)

    n <- dim(train.data)[1]
    nt <- dim(test.data)[1]
    class <- train.data[,1]
    xx <- train.data[,-1]

    #-----------------------------------------------------
    # Finds mean vectors for each class if the class is
    # represented in the training data.
    #-----------------------------------------------------

    for (y in 1:J){
       ind <- which(class == y)
       # Class y is not represented in the training data
       if (length(ind) == 0){
          mu[y,] <- rep("NA", p)
       }
       else{
          mu[y,] <- apply(xx[ind,], 2, mean)
       }
    } # END of for

    # finds the diag covariance matrix
    tmp <-var(xx)
    diag(sigma) <- diag(tmp)
    sigma.inv <- solve(sigma)
    # --------------------------------------------
    # Finds the linear discriminant function for
    # each case in test.data. This will later be
    # used for predicting the class label of
    # cases in the test set. If the class are not
    # represented in the training set, it will be
    # excluded from the prediction.
    #---------------------------------------------

    dlda.fun <- matrix(0, nrow = nt, ncol = J)
    for (i in 1:nt){
       x <- test.data[i,]
       for (y in 1:J){
          if ( mu[y,1] == "NA"){
             dlda.fun[i, y] <- -99999999999
          }
          else{
             dlda.fun[i,y] <- 2*log(pi[y],exp(1))-
                    ((x-mu[y,])%*%sigma.inv%*%(t(x-mu[y,])))
          }
       } # END of y
    } # End of i

    # Predict the class
    pred.class <- max.col(dlda.fun)
    pred.class

}# END of DLDA
```

```
#-----------------------------------------------------------------
#         Diagonal Quadratic Discriminant Analysis
#         ----------------------------------------
#  Diagonal QDA: assumes the class densities all have different
#                diagonal covariance matrix. The off-diagonal
#                are set to zero.
#  Pre:
#  ====
#              Pi: prior probabilities provided by the user
#      train.data: training data
#       test.data: test data
#               J: number of classes in training data
#               p: number of features
#  Post:
#  =====
#       Computes the mean vector and the diag covariance
#       matrix from the train data and predict the class for
#       each case in the test set.
#-----------------------------------------------------------------

DQDA <- function(pi, train.data, test.data, J, p)
{
    mu <- matrix(0, nrow = J, ncol = p)
    sigma <- array(0, dim=c(J, p, p))
    sigma.inv <- array(0, dim=c(J, p, p))

    n <- dim(train.data)[1]
    nt <- dim(test.data)[1]
    class <- train.data[,1]
    xx <- train.data[,-1]
```

```
#-----------------------------------------------------
# Compute the mean vectors and covarianfor each class
# if the class is represented in the training data.
#-----------------------------------------------------
 for (y in 1:J){
    ind <- which(class == y)
    if (length(ind) == 0){
       mu[y,] <- rep("NA",p)
    }
    else{
       mu[y,] <- apply(xx[ind,], 2, mean)
       temp <- var(xx[ind,])
       diag(sigma[y,,]) <- diag(temp)
       sigma.inv[y,,] <- solve(sigma[y,,])
    }
 } # END of y
 # ------------------------------------------------
 #  Finds the discriminant function for each x
 #  in test.data
 #-------------------------------------------------
 DQDA.fun <- matrix(0, nrow = nt, ncol = J)
 for (icase in 1:nt){
    x <- test.data[icase,]
    for (y in 1:J){
       if ( mu[y,1] == "NA"){
          DQDA.fun[icase,y] <- -999999999999
       }
```

```
          else{
              DQDA.fun[icase,y] <- (2*log(pi[y],exp(1))) -
                  ((x-mu[y,])%*%sigma.inv[y,,]%*%t(x-mu[y,]))
          }
       } # END of y
    } # END of icase
    pred.class <- max.col(DQDA.fun)
    pred.class
} # END OF DQDA


reptd.cv <- function(J, ncv, datafile, B, pi, cvindex, proc)
#------------------------------------------------------------
# reptd.cv : repeatedly trains "proc" B times on the data
#           " datafile". Estimates the misclassification
#           rates for each run using ncv-fold cross
#           validation.
# cvindex: an B by n matrix. It holds indexes to randomly
#          partition the data.
#       J: Number of classes.
#      pi: Vector of prior probabilities.
#------------------------------------------------------------
{
    pmc <- rep(0,B)                   # holds pmc for DLDA
    n <- dim(datafile)[1]             # number of cases
    p <- (dim(datafile)[2]) -1        # number of features
    k <- round(n/ncv)                 # size of cv Partitioning
```

```
for (b in 1:B){

   indx <- cvindex[b,]

   #-------------------------------------------

   #     ncv -fold cross validation begins here

   #-------------------------------------------

   pmcv <- rep(0,ncv)  # holds pmc for each cv

   for (icv in 1:ncv){

      if ( icv != ncv) {

         testindx <- indx[((icv-1)*k+1):(icv*k)]

         trnindx <- indx[-(((icv-1)*k+1):(icv*k))]

      }

      else {

         testindx <- indx[((icv-1)*k+1)  : n]

         trnindx <- indx[-(((icv-1)*k+1): n)]

      }

      Ytest <- datafile[testindx,1]

      ny.test <- tabulate(Ytest,J)      # Number of cases/class
                                        # in the test set
      testset <- datafile[testindx,-1]  # exclude the class label
                                        # from the testset
      trainset <- datafile[trnindx,]

      Ytrain <- datafile[trnindx,1]

      ny.train <- tabulate(Ytrain, J)

      pi <- ny.train/sum(ny.train)

      pred.class <- proc(pi,trainset,testset,J,p)


      # ---------------------------------------------

      # confusion matrix: row = true, col = predicted
```

```
            #--------------------------------------------------

            confus <- matrix(0, ncol=J, nrow=J)
            temp<- table(Ytest, pred.class )
            ny.pred <- tabulate(pred.class, J)
            ind1 <- which(ny.test != 0)
            ind2 <- which(ny.pred != 0)
            confus[ind1,ind2] <- temp
            # Estimation of pmc
            pcc <- sum(diag(confus))/sum(ny.test)
            pmcv[icv] <- 1 - pcc
        } # cv loop
        # Average pmcv
        pmc[b] <- mean(pmcv)
    } # END OF B
    pmc
} # END OF reptd.cv
```

Then, for example run DLDA 150 times on lymphoma data set, we can run:

```
    cvindex <- t(samp.permute(80,150))
    pmc.DLDA <- reptd.cv(3, 3, lymphoma, 150, cvindex, DLDA)
```