HIGH-DIMENSIONAL SIGNAL PROCESSING AND STATISTICAL LEARNING

Daniyar Bakir, Bachelor of Engineering

Submitted in fulfillment of the requirements for the degree of Master of Science

School of Engineering
Department of Electrical and Electronic Engineering
Nazarbayev University

53 Kabanbay batyr Avenue,
Astana, Kazakhstan, 010000

December 9

Abstract

Classical statistical and signal processing techniques are not generally useful in situations wherein the dimensionality (p) of observations is comparable or exceeding the sample size (n). This is mainly due to the fact that the performance of these techniques is guaranteed through classical notion of statistical consistency, which is itself fashioned for situations wherein $n >> p$. Statistical consistency has been viogorously used in the past century to develop many signal processing and statistical learning techniques. However, in recent years, two sets of mathematical machineries have emerged that show the possibility of developing superior techniques suitable for analyzing high-dimensional observations, i.e., situations where $p >> n$. In this thesis, we refer to these techniques, which are grounded either in double asymptotic regimes or sparsity assumptions, as *high-dimensional* techniques.

In this thesis, we examine and develop a set of high-dimensional techniques with applications in classification. The thesis is mainly divided to three parts. In the first part, we introduce a novel approach based on double asymptotics to estimate the regularization parameter used in a well-known technique known as RLDA classifier. We examine the robustness of the developed approach to Gaussianity, an assumption used in developing the core estimator. The performance of the technique in terms of accuracy and efficiency is verified against other popular methods such as cross-validation. In the second part of the thesis, the performance of the newly developed RLDA and several other classifiers are compared in situations where p is comparable or exceeding n.

While in the first two parts of the thesis, we focus more on double asymptotic methods, in the third part, we study two important class of techniques based on sparsity assumption. One of these techniques known as LASSO has gained much attention in recent years within the statistical community, while the second one, known as compressed sensing, has become very popular in signal processing literature. Although both of these techniques use sparsity assumptions as well as L_1 minimization, the objective functions and constrains they are constructed on are different. In the third part of the thesis, we demonstrate the application of both techniques in high-dimensional classification and compare them in terms of shrinkage rate and classification accuracy.

Acknowledgments

I would like to express my sincere appreciation to my advisers Dr. Amin Zollanvari and Dr. Alex P. James for their intensive guidance throughout my studies. A lot of thanks to Dr. Amin Zollanvari for his thorough support and teaching he gave me in the last two years. His support and advice were my guiding light directing me to a successful completion of my current studies. Many thanks to Dr. Alex P. James for supervising me throughout the last several years, his dedication to teach and support helped me in getting the most valuable knowledge I have today. It was mostly him who induced my interest in studying algorithms, data analysis and signal processing.

I would like to thank my friends and relatives for their sincere help and support in my life. Also, I would like to give my special thanks to my mother, Dana. Her love and kind words are the best cure for any obstacle or difficulty I faced and will face in my life. I have an unplayable debt to her, and hopefully as my return I will keep her always smiling and happy.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

CV Cross Validation

DLDA Diagonal Linear Discriminant Analysis

EDC Euclidean Distance Classifier

FOTT First-Order Tree Type

G13,G_{13} Girko Classifier

LDA Linear Discriminant Analysis

loo Leave-One-Out

RLDA Regularized Linear Discriminant Analysis

SERD,Serd Serdobolskij Classifier

SVM Support Vector Machine

ZAR,Zar Zarudskij Classifier

Chapter 1 – Introduction

Trying to predict an outcome from the past observations was always fascinating humankind in many problems such as everyday life weather forecast, currency exchange rates or medical classification based on gene expression profiles. The mathematical field that accomplishes such tasks is called pattern recognition or machine learning. The set of machineries used in pattern recognition can be divided into three intertwined parts: error estimation, feature extraction, and classification [1]. Giving a thorough discussion on each of them is not a feasible task, but readers may refer to [2–5] for more information. What concerns us here in this thesis is classification problems and their applications in high-dimensional settings, i.e., situations where the dimensionality of observations (the number of variables) is comparable or exceeding the sample size.

Throughout the last century, the classification rules were developed relying on a classical statistical notion of statistical consistency which is shaped under assumptions that number of observations (n) increases unboundedly while observation dimensionality (p) is held fixed [1]. Many classification techniques, known as classifiers, have been developed under such assumptions and the fact that these assumptions guarantee that classifiers should converge to the optimum classifier (Bayes classifier) in very large sample settings has been a tempting idea for many practitioners to use them. In contrary to the idea behind these methods, the modern databases have much more variables (features) compared to the available sample size. For example, consider an image processing application where each pixel can be viewed as one feature, while the amount of sample images of the object is strictly limited; or in gene microarray-based classification of phenotypes where each gene (out of tens of thousands genes) is a potential feature, while the number of available subjects (individuals) is very limited. Analyzing such datasets using classical techniques requires dimensionality reduction procedures that transforms the original high-dimensional data into a new low dimensional feature space. Another major factor that repulses community from using high-dimensional settings is related to the "curse of dimensionality" phenomenon (also known as the "peaking phenomenon"), which states that for a given sample size adding more feature variables to a classifier improves the predictive capacity only up to a certain point, after which the performance starts to deteriorate [6].

A solid example of the classifier operating in high-dimensions outperforming the one that use classical notion of statistical consistency is given in [1], where Euclidean Distance Classifier (EDC, will be discussed later) was used on a multinomial Gaussian distributed synthetic dataset with various configurations. As stated in [1], suppose $\boldsymbol{\theta}_0 = -\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_0 = [0.2^T_{(10)}, 0.05^T_{(190)}, 0.03^T_{(1200)}, 0^T_{(300)}]^T$ and $\boldsymbol{\Sigma} = \mathbf{I}_{1700}$, where $\mathbf{I}_p$ is a p dimensional identity matrix. As it is a common (e.g., see [7]), the "best" features were added first to the classifier and the accuracy of classifier was examined to see the validity of the peaking phenomenon. This choice of $\boldsymbol{\theta}_i$ also provides an opportunity to extract features easily since larger mean values of features more discriminative and will be picked first. The training data has 200 samples in total with $n = 100$ observations for each class. The expected true error of the EDC classifier is given in [8]:

$$E[\epsilon_{n,p}] \approx \Phi\left(\frac{-\delta_p^2}{\sqrt{\delta_p^2 + 2J}}\right), \tag{1.1}$$

where $J = p/n$, and

$$\delta_p^2 = (\boldsymbol{\theta}_0 - \boldsymbol{\theta}_1)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\theta}_0 - \boldsymbol{\theta}_1), \tag{1.2}$$

is known as Mahalanobis distance. It is notable that the optimum true error is obtained when $p = 1400$ ($E[\epsilon_{100,1440}] = 0.254$), while feeding all 1700 features to the model results in a lower misclassification rate than the first local minima ($E[\epsilon_{100,1700}] = 0.273 > E[\epsilon_{100,10}] = 0.276$). This means that using classifiers in high-dimensional setting would not certainly result in poor performance, instead it can reveal new unexpected findings.

Exploring high-dimensional space using tools that were originally designed for classical statistical assumptions is not always probable. EDC being a modification of popular Linear Discriminant Analysis (LDA, will be discussed later), can classify data in any positive p/n ratio ($p/n > 0$), while its original counterpart LDA is not defined when data dimensionality reaches the sample size ($p/n \geq 1$). This implies that scientists should not rely (but still consider) on classical statistical notion when looking for classifiers that operate in high-dimensions and shift towards new frameworks in designing high-dimensional classifiers.

One of the ways to design a high-dimensional classifier is to apply the Girko analysis (general statistical analysis of observations also known as G-analysis) [9], which deploys double asymptotic assumptions $n \to \infty$, $p \to \infty$, $p/n \to c$, where $0 < c < \infty$. This framework is also the groundwork of the Random Matrix Theory (RMT) that has been used successfully in recent years in wireless communications [10]. Nevertheless, not much research has been conducted on applications of double asymptotics in machine learning area [1], despite its usage is well justified by numerous examples from other fields such as nuclear physics and signal processing (see [1, 11]).

Another technique to design a classifier for high-dimensional analysis is called shrinkage. Generally, shrinkage methods are useful when there is a sparsity of the model, i.e., only a certain number of feature variables account for the response variable [12]. Unlike the methods that use explicit feature extraction procedure, the process of feature extraction remains implicit in shrinkage-based techniques [13].

The relevance of high-dimensional classification, whether based on double asymptotics or shrinkage, lies in a continuously increasing amount of information that is needed to be analyzed. For example, in medicine, studying many genes simultaneously is helpful in classifying and revealing new types of cancer [14], drug development [15], or estimating cancer survival rate according to the probability of cancer relapse (see [16] for more information). This thesis discusses the mathematical tools that stand behind these discoveries, particularly: Chapter 2 introduces a new methodology to estimate the regularization parameter of Regularized Linear Discriminant Analysis (RLDA). In Chapter 3 one can find a comprehensive performance comparison of several classifiers designed under classical and double asymptotic assumptions, and Chapter 4 details two classification schemes based on model sparsity assumptions.

Throughout the report, a uniform mathematical consistency of variables is used. A bold greek or latin symbol written in lowercase represents a column vector, e.g., $\mathbf{x}$ or $\boldsymbol{\beta}$; a bold capital letter stands for matrix, e.g., $\mathbf{A}$; $\mathbf{I}_p$ stands for the p dimensional identity matrix; a super index T, e.g., $\mathbf{x}^T$, denotes the transpose operation; $||\mathbf{x}||_2$ indicates the L_2 norm distance; and finally, tr[.] is a trace operator (sum of diagonal elements).

Chapter 2 – An Efficient Methodology of Estimating Regularization Parameter in RLDA

2.1 Introduction

Linear Discriminant Analysis (LDA) is a popular classification scheme that can be applied in situations where the dimensionality of observations is less than the sample size, i.e., when $p < n$. In situations where $p \geq n$, one of the building blocks of the LDA (the covariance matrix) becomes ill-conditioned, and as a result LDA is not defined. To overcome that Di Pillo [17] replaced the ill-conditioned inverse estimated covariance matrix of LDA with a modified one that is stabilized by introducing a regularization parameter, and constructed a classifier known as the Regularized LDA (RLDA). Di Pillo based his work on the idea of Hoerl and Kennads who stabilized the ill-conditioned ridge regression suffering from the same problem [18–20]. However, as stated by Di Pillo, careful selection of the regularization parameter is important because it can substantially change the performance of RLDA [21]. According to Peck and Ness [22], Di Pillo came to the conclusion that the analytical solution of the optimum regularization parameter (γ) in RLDA is intractable and in practice it should be estimated from empirical observations.

Currently, popular cross-validation (CV) techniques can be deployed to estimate the optimum γ; however, due to repetitive nature of CV, the classification rule is applied on training data for each value of γ during the search and repeated several times, thereby this approach is not computationally efficient. In this chapter, we introduce an efficient methodology to estimate the regularization parameter in RLDA and compare the performance of the estimation technique with conventional techniques that use plug-in estimator or CV schemes such as 5 fold, 5 repetitions CV (CV5F-5R) and leave-one-out CV (LOO). The new methodology is based on a recently developed RLDA true error estimator developed in [23] and computes the expected true error with a pre-defined exponential range of γ, after which the γ with the lowest error can be estimated.

2.2 Methodology

LDA was firstly introduced by Ronald. A. Fisher in 1936 [24] to classify different plants in taxonomic data. Fisher founded his idea based on maximizing the ratio of between class to within class scattering matrices. The same principle is used in the modern LDA but with an assumption of the common covariance matrix between classes. Since the LDA was first introduced in 1936s, it has found many applications in face recognition [25], cancer genomics [26], finance problems [27], and others.

Consider a binary classification problem with common covariance matrix [28]. The samples for each class, $\mathbf{X}_i = \{\mathbf{x}_{in_i}\}$ are driven with a class mean value $\bar{\mathbf{x}}_0$ and $\bar{\mathbf{x}}_1$ with p amount of features, n_0 and n_1 samples for classes 0 and 1 respectively. Let the difference between the mean vectors $\mathbf{d} = \mathbf{x}_0 - \mathbf{x}_1$, the sample covariance matrix

$$\mathbf{S} = \frac{1}{n_0 + n_1 - 2}\mathbf{A},$$

where

$$\mathbf{A} = \sum_{i=0,1} \sum_{j=1}^{n_i} (\mathbf{x}_i - \bar{\mathbf{x}}_i)(\mathbf{x}_i - \bar{\mathbf{x}}_i)^T .$$

The arbitrary linear combination

$$Z = \mathbf{b}^T \mathbf{x},$$

the its difference between the sample means

$$\bar{\mathbf{Z}}_0 - \bar{\mathbf{Z}}_1 = \mathbf{b}^T \mathbf{x}$$

and its variance is

$$\operatorname{var}(\mathbf{Z}) = \mathbf{b}^T \mathbf{S} \mathbf{b}.$$

Original LDA (Fisher's LDA) objective was to maximize the ratio

$$\frac{(\bar{\mathbf{Z}}_0 - \bar{\mathbf{Z}}_1)^2}{\operatorname{var}(Z)} = \frac{(\mathbf{b}^T \mathbf{d})^2}{\mathbf{b}^T \mathbf{S} \mathbf{b}}$$

with respect to $\mathbf{b}$. The solution to this problem is

$$\mathbf{b} = \mathbf{S}^{-1} \mathbf{d}.$$

The classification rule ($\mathbf{b}^T \mathbf{x} = \frac{1}{2} (\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1)$ can be assigned to either class) is given by

$$\psi(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{b}^T \mathbf{x} > \frac{1}{2} (\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1) \\ 1 & \text{if } \mathbf{b}^T \mathbf{x} \leq \frac{1}{2} (\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1) \end{cases}$$

Modern LDA (LDA hereafter) classifier is grounded in the following assumptions [29]. Suppose a binary classification problem in which data from class i ($i = 0, 1$) follows a multivariate Gaussian distribution $\mathbb{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$ for $i = 0, 1$. Note that $\boldsymbol{\Sigma}$ does not depend on i and is identical across both classes. The sample space S_0 and S_1 with n_0 and n_1 amount of samples are driven from $\mathbb{R}^p$ populations Π_0 and Π_1 respectively. Let $n = n_0 + n_1$ and n_0, n_1 being predefined constants, i.e., case of separate sampling is considered. The true error of classifier, ε, is defined as

$$\varepsilon = \alpha_0\varepsilon_0 + \alpha_1\varepsilon_1, \tag{2.1}$$

where α_i is a prior probability of class i and ε_i is the misclassification rate the classifier commits on future sample point coming from class i. Since α_i's are unknown in many cases, they are estimated as $\alpha_i = n_i/n$, which converge to the population values as number of observations increases unboundedly. The modern LDA uses the same principle as was described by R. Fisher in [24], but with the assumption of Gaussianity of data with a common covariance matrix across classes, it can be represented as Anderson's statistics [29] given by:

$$W^{LDA}(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{C}, \mathbf{x}) = \left(\mathbf{x} - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T \mathbf{C}^{-1} \left(\mathbf{x}_0 - \mathbf{x}_1\right), \tag{2.2}$$

where $\bar{\mathbf{x}}_i = \frac{1}{n_i}\sum_{\mathbf{x}_l \in S_i} \mathbf{x}_l$ is sample mean for class i, $\mathbf{C}$ is the pooled sample covariance matrix,

$$\mathbf{C} = \frac{(n_0 - 1)\mathbf{C}_0 + (n_1 - 0)\mathbf{C}_1}{n_0 + n_1 - 2}, \tag{2.3}$$

with

$$\mathbf{C}_i = \frac{1}{n_i - 1}\sum_{\mathbf{x}_l \in S_i} (\mathbf{x}_l - \bar{\mathbf{x}}_i)(\mathbf{x}_l - \bar{\mathbf{x}}_i)^T. \tag{2.4}$$

The RLDA discriminant modifies the inverse pooled sample covariance matrix used in LDA by defining,

$$W^{RLDA}\left(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{C}, \mathbf{x}\right) = \kappa \left(\mathbf{x} - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T \mathbf{H}\left(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\right), \tag{2.5}$$

where $\kappa > 0$, $\gamma > 0$ and,

$$H = \left(\mathbf{I}_p + \gamma \mathbf{C}\right)^{-1}. \tag{2.6}$$

The classification rule for LDA and RLDA is given by

$$\psi_n(\mathbf{x}) = \begin{cases} 1, & \text{if } W \leq c \\ 0, & \text{otherwise} \end{cases}, \tag{2.7}$$

where $c = \log \frac{1-\alpha_0}{\alpha_1}$ and W indicates the discriminant functions given by (2.2) or (2.5) for LDA or RLDA, respectively. The generalized consistent estimator introduced in [23] provides an analytical expression to estimate the true error that an RLDA classifier commits on class i as:

$$\hat{\varepsilon}_i^D = \Phi\left(\frac{(-1)^{i+1} G\left(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}\right) + \frac{(n_0+n_1-2)\hat{\delta}}{n_i} + (-1)^{-i} \frac{c}{\kappa}}{\sqrt{\left(1 + \gamma\hat{\delta}\right)^2 D\left(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}, \mathbf{C}\right)}}\right), \tag{2.8}$$

where

$$\hat{\delta} = \frac{\frac{p}{n_0+n_1-2} - \frac{\text{tr}[\mathbf{H}]}{n_0+n_1-2}}{\gamma\left(1 - \frac{p}{n_0+n_1-2} + \frac{\text{tr}[\mathbf{H}]}{n_0+n_1-2}\right)}, \tag{2.9}$$

while

$$G\left(\boldsymbol{\mu}_i, \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}\right) = \left(\boldsymbol{\mu}_i - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T \mathbf{H}\left(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\right), \tag{2.10}$$

and

$$D(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}, \mathbf{C}) = (\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1)^T \mathbf{HCH} (\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1). \tag{2.11}$$

Then the overall estimated error of RLDA is

$$\hat{\varepsilon}^D = \alpha_0 \hat{\varepsilon}_0^D + \alpha_1 \hat{\varepsilon}_1^D. \tag{2.12}$$

where $\hat{\varepsilon}_0^D$ is defined in (2.8). This estimator is based on the concept of generalized consistent estimation, which is itself based on double asymptotic framework (see [23] for more information). Later in this chapter, we employ the estimator in a one-dimensional range search to estimate the optimum regularization parameter and compare the performance of estimation to cross-validation (CV5F-5R and leave-one-out) and plug-in estimator. For a sample data of size n, the leave-one-out estimation technique is summarized below:

Step 1: Set $j = 0$

Step 2: Set aside one of the sample points from the training data

Step 3: Use all other sample points to train the classifier

Step 4: Apply the classifier (in our case RLDA) to classify the held-out sample point

Step 5: Increment j if the held-out sample point is misclassified

Step 6: Repeat Steps 2-4 for any sample point that has not been held out yet. Go to the next step when all sample points have been held out once.

Step 7: Estimate the true error as j/n

To find the γ^{opt} using leave-one-out (loo), we need to estimate the true error for each γ and trace the γ value that corresponds to the minimum error estimate.

The CV5F-5R is similar to leave-one-out with a difference in the number of held-out samples and repetitions to perform. As the name suggests, CV5F-5R breaks the training data to 5 folds and repeats the main process 5 times. The steps to conduct CV5F-5R are given below:

Step 1: Randomly divide the training data into 5 folds

Step 2: Set aside one fold

Step 3: Use other folds to train the classifier

Step 4: Apply the trained classifier to the held-out sample and estimate the error

Step 5: Repeat steps 3 and 4 and for any fold that has not been held out yet

Step 6: Repeat all previous steps 5 times

Step 7: Estimate the true error as the average of all estimated errors

The γ^{opt} is the one that corresponds to the least estimated error using CV5F-5R.

The plug-in estimator was derived in [30] and is given by

$$\hat{\varepsilon}_i^P = \Phi\left(\frac{(-1)^{i+1}\, G\left(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{x}\right) + (-1)^{-1}\,\frac{c}{\kappa}}{\sqrt{D\left(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}, \mathbf{C}\right)}}\right), \tag{2.13}$$

This estimator has the property that under classical large sample assumptions ($n \to \infty$ and p is fixed) it converges to the true error of RLDA (statistical consistency). This is different from (2.8) that converges to true error under double asymptotic assumptions, $n \to \infty$, $p \to \infty$, $p/n \to c$, where $0 < c < \infty$.

In order to compare the performance of the aforementioned estimation techniques in estimating the optimum value of regularization parameter used in RLDA, we conduct a set of simulations and examine which estimator results in the smallest average true error. This means that in each scenario that we consider (i.e., for each n, p, and each dataset), we need to have a knowledge of the true error of the constructed classifier as well. In simulations using real datasets, we have no knowledge of class conditional probability distributions and the true error *per se* needs to be estimated from the data in hand. This can be done via the hold-out estimator, i.e., randomly divide the full dataset into two different sets, the training and testing sets. Estimate γ^{opt} and train the RLDA classifier using the training set, and apply the classifier on test data to estimate the true error. Repeating this process many times and taking the average of the estimated hold-out error converges to the expected true error of the classifier conditional on n, p, and the dataset.

We have deployed Monte-Carlo simulations using synthetic and real datasets to study the performance of the aforementioned estimators, namely, CV5F-5R, loo, plug-in ($\hat{\varepsilon}^P$), and $\hat{\varepsilon}^D$ (in figures identified by "dasym"), in estimating γ^{opt}. In experiments using real datasets, the initial large number of features has been reduced to a number comparable to the sample size. This was done by a two sample t-test and selecting those features which the least p-values. In our simulation study, we have used the following protocol in experiments using real data:

Protocol 2.1:

Step 1: Let $r = N_0/N_1$ express the ratio of the number of observations in class 0 and class 1, and $N = N_0 + N_1$ be the total number of samples in the data. Select a set of observations of size n ($n < N$) and generate a training set such that the proportion of sample points from either class follows the value of r obtained from the full dataset. Let $n_0 = \lfloor rn_1 \rfloor$, where $\lfloor . \rfloor$ indicates the floor function. Setting $n = n_0 + n_1$ and having n_0 leads to $n_1 = \lfloor \frac{n}{r+1} \rfloor$. Randomly select a set of training data of size $n = \{30, 40, ..., 100\}$ and set aside the rest for testing.

Step 2: Let $\gamma = \gamma^i_{\text{base}}$ for $i = \{-10, -9, ..., 0, 1, ...10\}$, where $\gamma_{\text{base}} = (1000)^{1/10}$. The choice of an exponential function for possible range of γ is justified because for small (large) values of γ a small change can have potentially a large (small) impact on the performance of RLDA. This exponential range of γ allows us to skip too many unnecessary computations in our search for optimal γ.

Step 3: For each value of γ determine CV5F-5R, loo, ε^D, and ε^P error estimates as well as the true error (via hold-out estimator).

Step 4: Find the γ^{opt} from the range of γ via given estimators. For each estimator, γ^{opt} is the γ that corresponds to the least value of the estimate in the pre-determined exponential range. Record the value of true error corresponding to γ^{opt} (the value of true error is available from the previous step).

Step 5: Repeat Steps **1**-**4** 500 times and determine the average expected error rate of the classifier.

The set of real datasets used in our experiments is provided in Table 2.1. All real databases were collected from [31], [32] and [33]. Description for each database is given in Appendix A.

Table 2.1: Microarray studies used in Chapter 2 experiments

Dataset	Features	n_0/n_1
Chen [34]	$10,237$	$75/82$
Desmedt [35]	$22,215$	$98/77$
Natsoulis [36]	$8,491$	$120/61$
Rosenwald [37]	$5,013$	$114/89$
Valk [38]	$22,215$	$116/157$
Vijver [39]	$5,003$	$180/115$
Yeoh [40]	$5,077$	$149/99$

The synthetic data used in our experiments have been generated by Gaussian and skewed-normal distributions. In experiments conducted using Gaussian distributed data, the sample size ranged from 30 to 300 and the number of features was $p = \{5, 20, 50, 150\}$. The protocol (pseudo-code for γ^{opt} estimation algorithm is given in D) for synthetic data experiments using Gaussian distributions is given below:

Protocol 2.2:

Step 1: Let $\mathbf{\Sigma}$ be 1 on the diagonal and 0.1 off the diagonal elements, $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_0$, where $\boldsymbol{\mu}_0 = (a, a, \ldots, a)$ and a is selected according to Mahalanobis distance, Δ, between classes $[\Delta^2 = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^T \mathbf{\Sigma}^{-1} (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)]$. By varying the Mahalanobis distance we change the lower bound on the true error rate of the classifier, also known as Bayes error. For $\Delta^2 = \{9, 5, 2, 0.75\}$ the Bayes error $\epsilon^{\text{Bayes}} = \{0.066, 0.131.0.239, 0.332\}$, respectively.

Step 2: Generate a set of samples of size n_0 and n_1 from populations $\Pi_0 = N(\boldsymbol{\mu}_0, \mathbf{\Sigma})$ and $\Pi_1 = N(\boldsymbol{\mu}_0, \mathbf{\Sigma})$ respectively, such that $n_0 = n_1 = \frac{n}{2}$. Note that in this case $\alpha_0 = \alpha_1 = \frac{1}{2}$.

Step 3: For each value of γ in a pre-determined exponential range, determine CV5F-5R, loo, ε^D, and ε^P error estimates. As before in our experiments we used $\gamma = \gamma^i_{\text{base}}$, where $\gamma_{base} = (1000)^{1/10}$, and $i = \{-10, -9, \ldots, 10\}$.

Step 4: Compute the true error of the classifier using (2.1) and (2.13) by replacing the sample parameters by their corresponding population values.

Step 5: Find the γ^{opt} from the range of γ via given estimators. For each estimator, γ^{opt} is the γ that corresponds to the least value of the estimate in the pre-determined exponential range. Record the value of true error corresponding to γ^{opt} (the value of true error is available from the previous step).

Step 6: Repeat Steps **1**-**5** 500 times and determine the average expected error rate of the classifier.

The skewed-normal ($\mathbf{z}\, SN(\boldsymbol{\mu}, \mathbf{\Sigma}, \boldsymbol{\beta})$) distribution is a modification of Gaussian distribution, but it adds the skewness factor to the overall data according to

$$2\phi_p(\mathbf{z}; \boldsymbol{\mu}, \mathbf{\Sigma})\, \Phi\left(\boldsymbol{\beta}^T (\mathbf{z} - \boldsymbol{\mu})\right), \tag{2.14}$$

where $\phi(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\Sigma})$ is a p dimensional normal density with mean vector of $\boldsymbol{\mu}$ and covariance matrix of $\boldsymbol{\Sigma}$, $\Phi\left(\boldsymbol{\beta}^T(\mathbf{z}-\boldsymbol{\mu})\right)$ is standard normal distribution, and $\boldsymbol{\beta} = \{\beta,\beta,\ldots\}$ is a p-dimensional "shape parameter vector" [41] which adds the skewness to the normal distribution. The real mean and covariance matrix of skewed-normal distribution are not $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, and letting $\beta = 0$ leads to a Gaussian distribution with such population parameters.

The experiments for skewed-normal and Gaussian distributions are very similar. We just replace the Gaussian distributions in Protocol 2.2 with skewed-normal distribution given in (2.14) with parameters $\beta = \{2,4\}$, and choose $n = \{30,40,\ldots,100\}$, and $p = \{20,50\}$. The values of $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_0$, where $\boldsymbol{\mu}_0 = (a,a,\ldots,a)$ is chosen such that the Mahalanobis distance, $\Delta = 2$, and $\boldsymbol{\Sigma}$ having 1 on the diagonal and 0.1 as the off diagonal elements.

2.3 Results and Discussion

Fig. 2.1 and 2.2 show the γ^{opt} for real databases when feature size $p = 50$ and $p = 150$, respectively. In these figures, each dataset listed in Table 2.1 is represented in a single row with four columns corresponding to various estimators and a single column corresponding to the true error itself (hold-out estimator). Each plot includes eight different curves corresponding to different training sample size. The vertical axis in each figure were scaled (except the plugin estimator) to facilitate the comparison of the performance of each estimator in estimating the actual γ^{opt}, which correspond to the γ with least true error in the right most column. It is seen that the plugin estimator's error has a monotone decreasing function as γ increases, and eventually converges to zero. This suggests that the

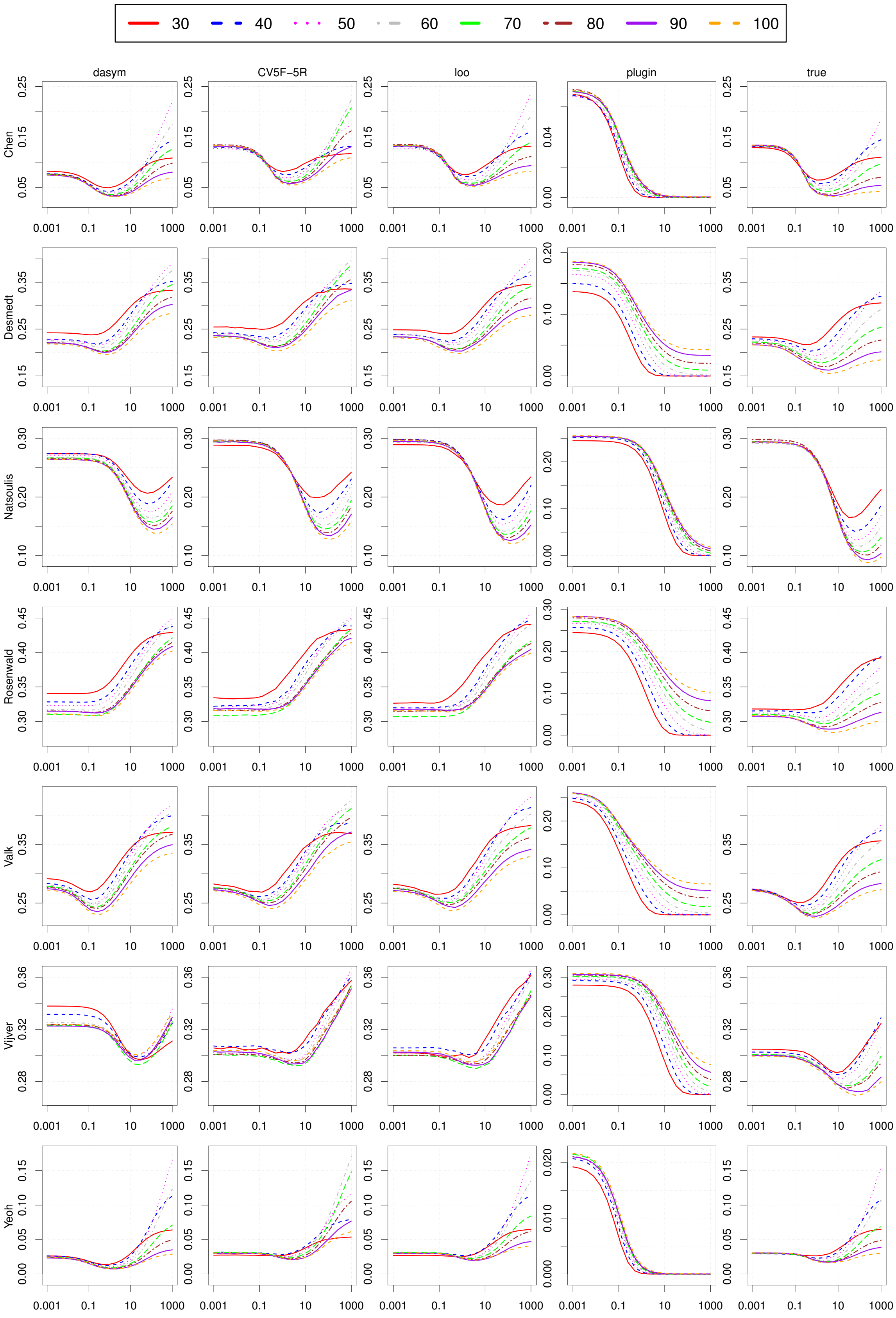


Figure 2.1: Expected estimated and true error (y-axis) versus regularization parameter $\log(\gamma)$ ***(x-axis) for several real datasets listed in Table 2.1 when*** $p = 50$***.***

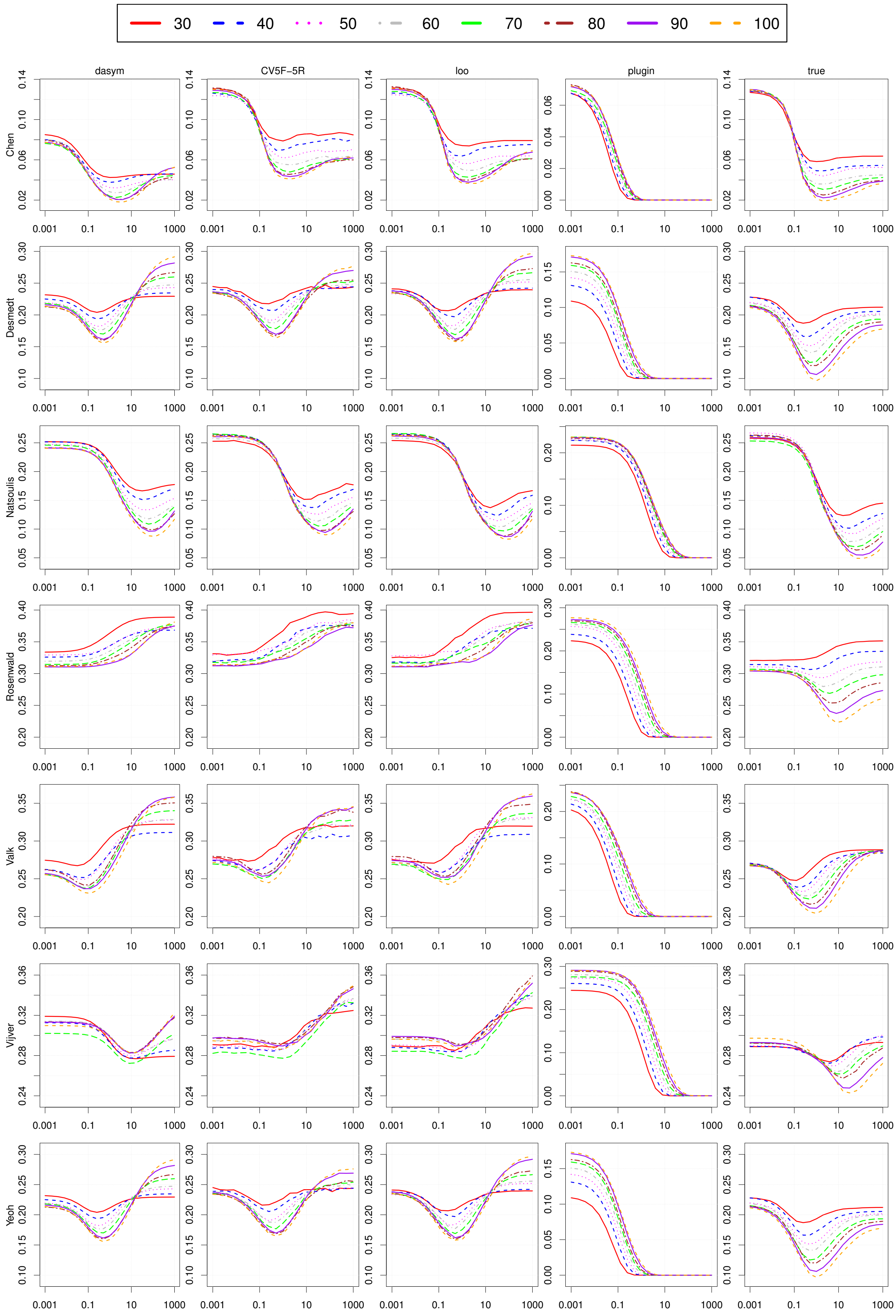


Figure 2.2: Expected estimated and true error (y-axis) versus regularization parameter $\log(\gamma)$ ***(x-axis) for several real datasets listed in Table 2.1 when*** $p = 150$***.***

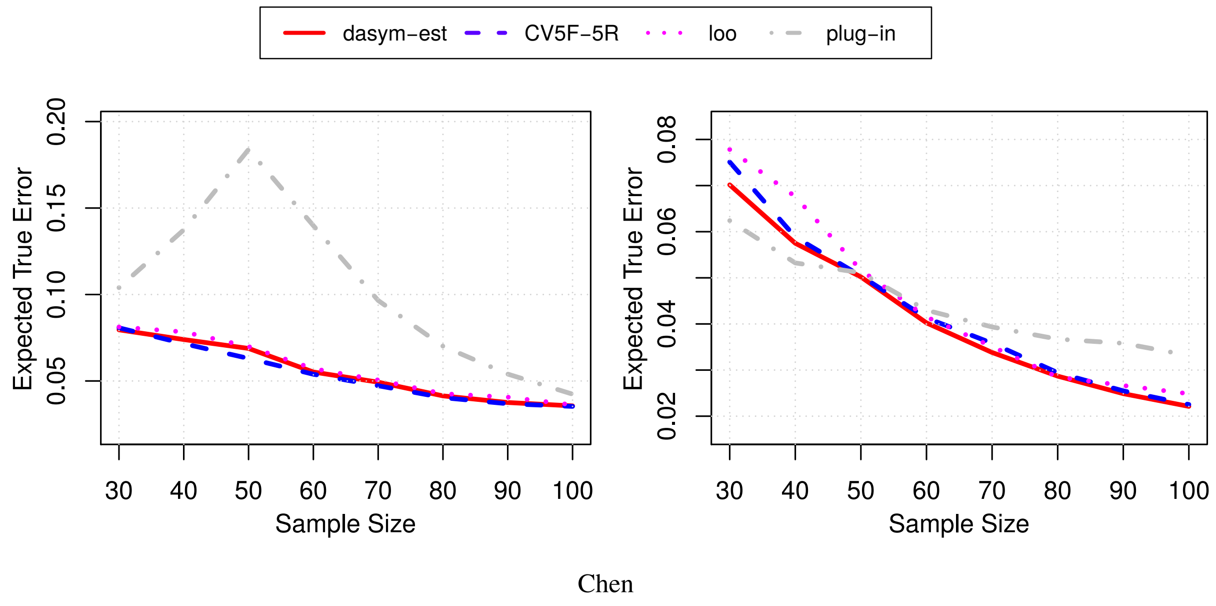


Figure 2.3: Expected true error as a function of sample size when $p = 50$ *(left) and* $p = 150$ *(right).*

plugin estimator is not a good estimator of regularization parameter (compare with the curves of true error in the right most column). All other estimators show the non-linear behavior of the expected true error as a function of the regularization parameter γ. Depending on the real data type, the γ^{opt} lies in a different range which is well identified by dasym-est, CV5F-5R, and loo estimators.

Figures 2.3 to 2.5, 2.6 and 2.7 show the expected true error rates with estimated γ^{opt} for real, Gaussian and skewed-normally distributed synthetic data respectively. Even though that at a relatively large sample size, the plug-in estimator shows a relatively good performance and sometimes outperforms other estimators (e.g., see 2.4c and 2.5c), it is in general inferior to other estimators. In all real databases, regardless of the training sample size and number of features, $\hat{\varepsilon}^{D}$ (dasym-est in figures) has a similar or better performance compared to CV estimators (loo or CV5F-5R). The result of experiments using synthetic data generated either from normal distributions (Fig. 2.6) or skewed-normal

Table 2.2: Time Calculations $p = 50$

Sample size	Total time for all iterations, s					Average time for each iteration, ms				
	$\hat{\varepsilon}^D$	CV5F-5R	loo	$\hat{\varepsilon}^P$	true	$\hat{\varepsilon}^D$	CV5F-5R	loo	$\hat{\varepsilon}^P$	true
30	3.8	159.1	81.4	3.4	56.6	7.5	318.2	162.8	6.8	113.2
40	3.8	172.3	109.0	3.4	53.5	7.6	344.7	218.0	6.7	107.0
50	3.6	179.5	136.1	3.3	47.1	7.2	359.1	272.1	6.6	95.5
60	3.6	196.2	161.3	3.3	47.1	7.2	392.4	322.6	6.6	90.9
70	3.6	209.1	189.7	3.3	45.5	7.3	418.3	379.4	6.7	94.2
80	3.7	223.1	217.6	3.3	44.3	7.3	446.2	435.2	6.7	88.7
90	3.6	236.1	245.7	3.3	42.8	7.3	472.2	491.4	6.6	85.5
100	3.7	249.7	274.6	3.3	41.2	7.3	499.4	549.1	6.5	82.3

distributions (Fig. 2.7) suggest a similar trend. Even though one of the primary assumptions in developing $\hat{\varepsilon}^D$ is the Gaussianity of data, good performance on skewed-normal distributions as well as real data demonstrate the robustness of $\hat{\varepsilon}^D$ to the non-Gaussian distributions.

Since the cross-validation estimators are based on constructing a set of surrogate classifiers and resampling procedure, the overall computational time of the range search for γ^{opt} is significant and increases as a function of sample size and the number of features. On the other hand, $\hat{\varepsilon}^D$ is calculated through a closed-form expression given in (2.8) avoiding highly computationally intensive procedure similar to those used in CV estimation. Tables 2.2 and 2.3 show the computational time spent to apply all estimators on one of the real datasets (Chen dataset [34]) when feature size $p = 50$ and $p = 150$, respectively. As the sample

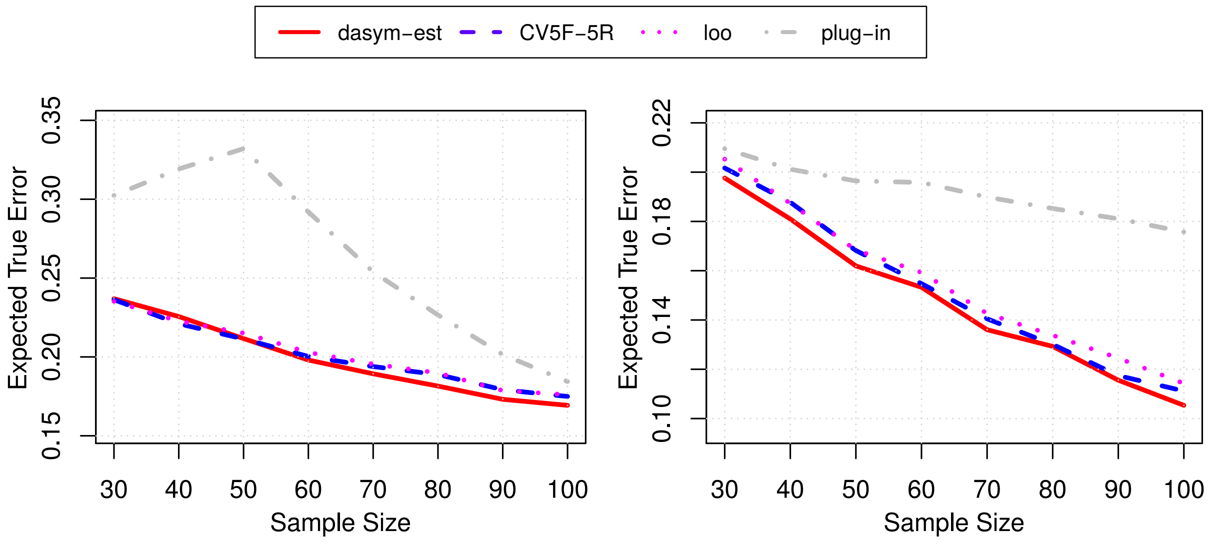
Table 2.3: Time Calculations $p = 150$

Sample size	Total time for all iterations, s					Average time for each iteration, ms				
	dasym-est	CV5F-5R	loo	plugin	true	dasym-est	CV5F-5R	loo	plugin	true
30	16.1	445.4	425.3	15.9	43.1	80.7	2227.0	2126.7	79.4	215.4
40	16.0	462.4	567.6	15.8	41.1	80.2	2312.1	2837.8	79.0	205.3
50	16.0	482.0	713.0	15.9	40.1	80.0	2409.9	3565.1	79.6	204.8
60	16.0	498.4	857.5	16.1	41.2	79.8	2492.2	4287.3	80.5	206.1
70	16.2	521.4	1006.3	16.4	40.4	80.9	r2607.1	5031.7	82.0	202.2
80	16.0	542.1	1156.2	15.9	41.1	80.1	2710.3	5780.8	79.6	205.6
90	16.2	565.8	1310.7	16.3	41.2	81.0	2828.8	6553.3	81.7	206.1
100	16.1	586.1	1459.7	16.0	41.4	80.7	2930.5	7298.3	80.0	207.2

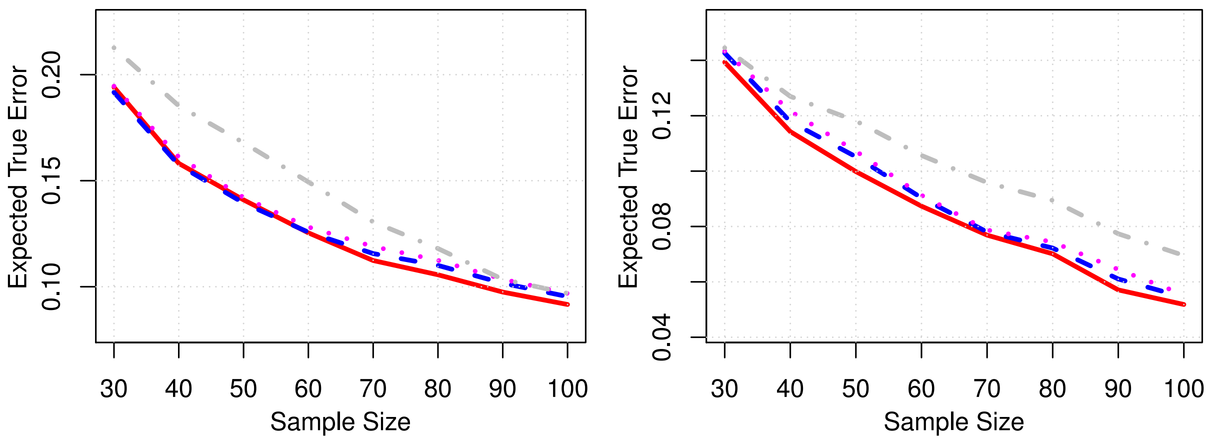
size and the number of feature increases, the CV-based search needs more time to estimate γ^{opt}, while the computational time of $\hat{\varepsilon}^D$-based search did not show any substantial change with larger sample sizes. This is also clear from Fig. 2.8, which shows the ratio of the average compute time of range searches based on CV and loo to $\hat{\varepsilon}^D$ and $\hat{\varepsilon}^P$. To summarize, the results show that the proposed $\hat{\varepsilon}^D$-based search of the optimum regularization parameter can potentially result in a RLDA classifier with a comparable or better accuracy than RLDA classifiers constructed using CV-based search scheme, while at the same time, being tens to hundreds of times faster to compute.

2.4 Conclusions

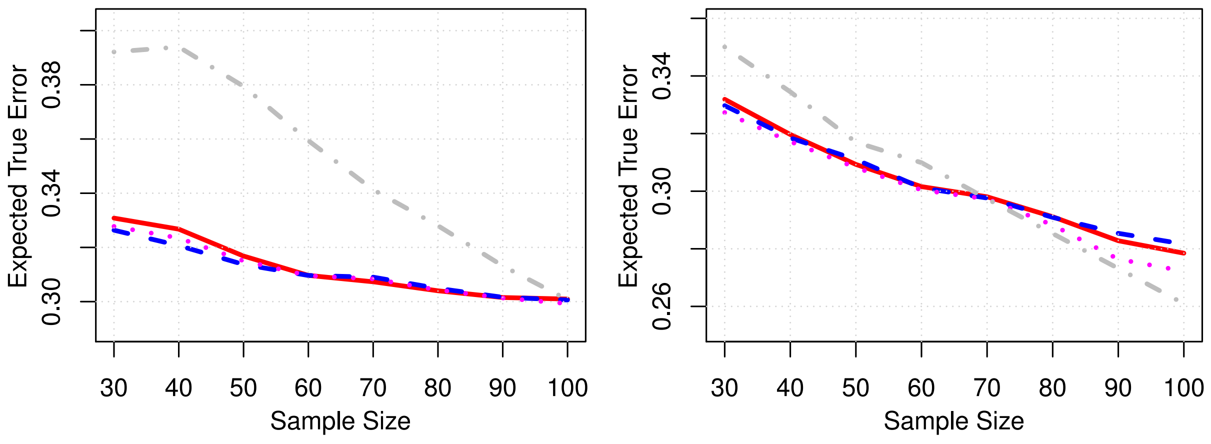
Efficient estimation of regularization parameter in RLDA is essential in high-dimensional settings, where LDA becomes degenerate and either cannot perform well or not defined at all. This chapter aimed to introduce a new methodology to estimate the optimum regularization parameter in RLDA by using the concepts of generalized consistent estimation. In this regard, a generalized consistent estimator of RLDA true error ($\hat{\varepsilon}^D$) is proposed to be used in estimating the optimum regularization parameter. This estimator is constructed using double asymptotic assumptions ($n \to \infty$, $p \to \infty$, $p/n \to c$, $0 < c < \infty$). The performance of a range search technique based on the estimator was compared to a similar search technique based on cross-validation procedures such as CV5F-5R (five folds, five repetitions) and leave-one-out (loo) as well as the simple plug-in estimator designed under classical assumptions used in statistics ($n \to \infty$, p is fixed). The proposed methodology is based on searching for the optimum γ from the exponential grid and its performance was verified on seven real databases and a set of synthetic data generated using Gaussian and skewed-normal distributions for various settings which represent both the low- and high-dimensional spaces. In all experiments $\hat{\varepsilon}^D$ shows a performance (expected true error) that is better or comparable to CV estimators. Having an analytical closed form expression and the absence of a resampling procedure in $\hat{\varepsilon}^D$ result in a low computational time. The results confirm that the proposed $\hat{\varepsilon}^D$-based search technique is tens to hundreds of times faster than a similar search technique that uses CV estimators. The downside of the search scheme lies in the pre-defined range of possible optimum γ. Future work needs to be done to better elucidate the possible range of γ^{opt}. Nevertheless, as it was mentioned before, analytical solution for the exact value of γ^{opt} depends on class conditional distributions, which are virtually unknown in practice, and, so might be the possible range of γ^{opt}.



(a) Desmedt

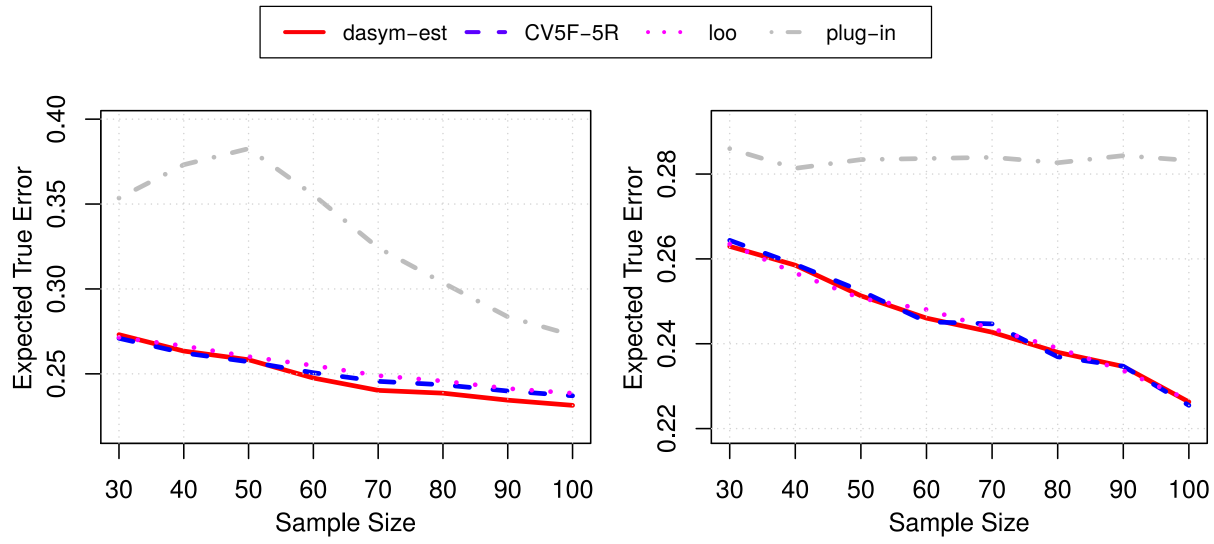


(b) Natsoulis

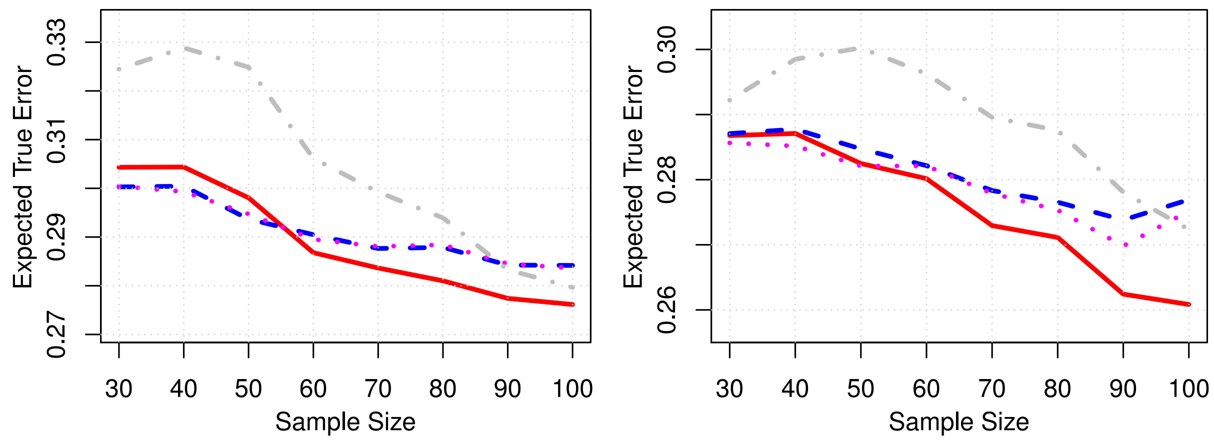


(c) Rosenwald

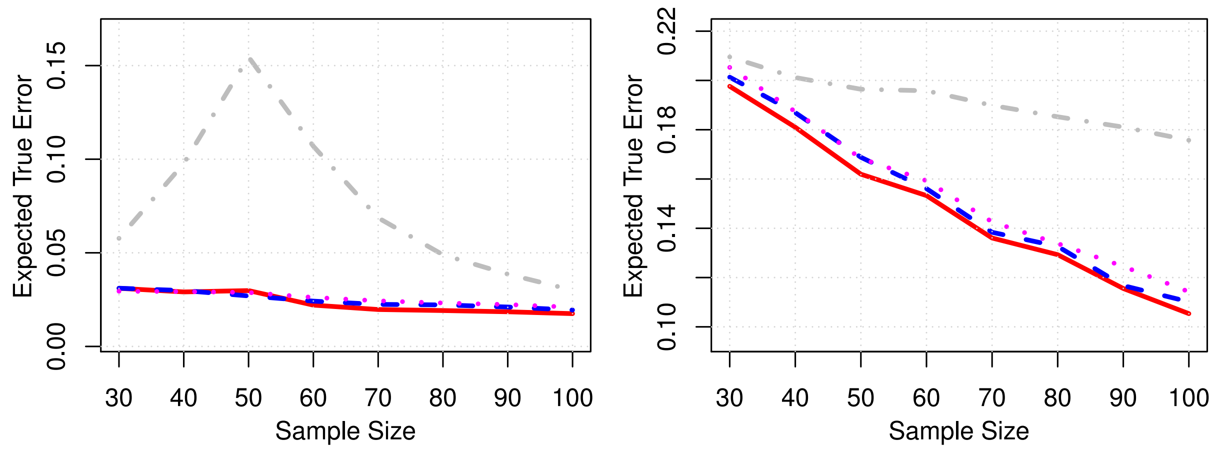
Figure 2.4: Expected true error as a function of sample size for several real datasets listed in Table 2.1 when $p = 50$ ***(left) and*** $p = 150$ ***(right).***



(a) Valk

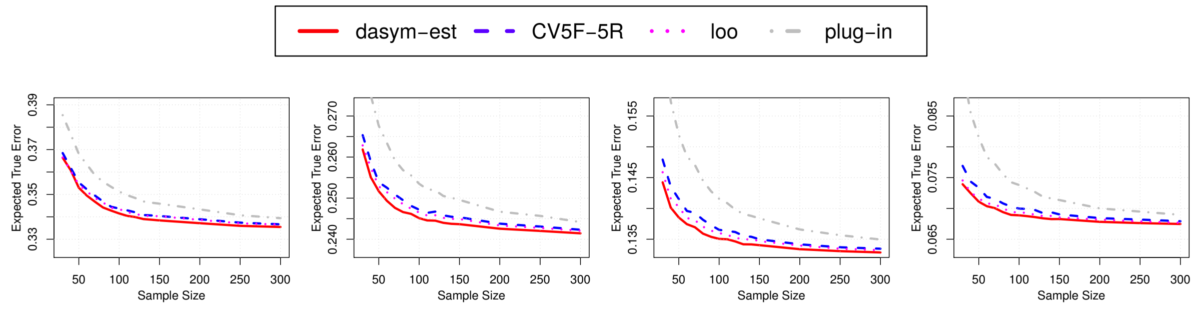


(b) Vijver

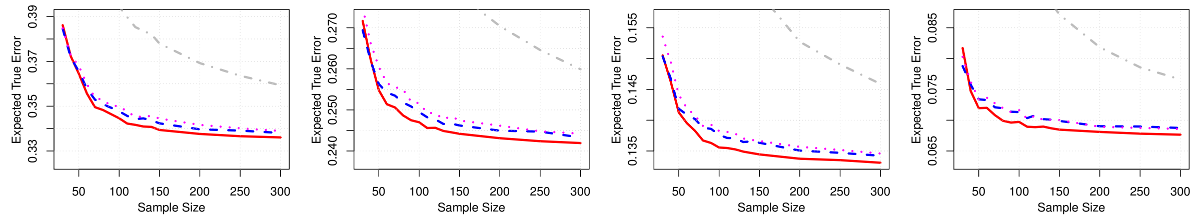


(c) Vijver

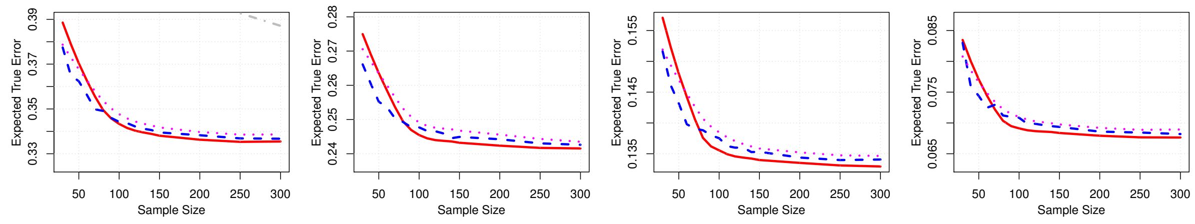
Figure 2.5: Expected true error as a function of sample size for several real datasets listed in Table 2.1 when $p = 50$ ***(left) and*** $p = 150$ ***(right).***



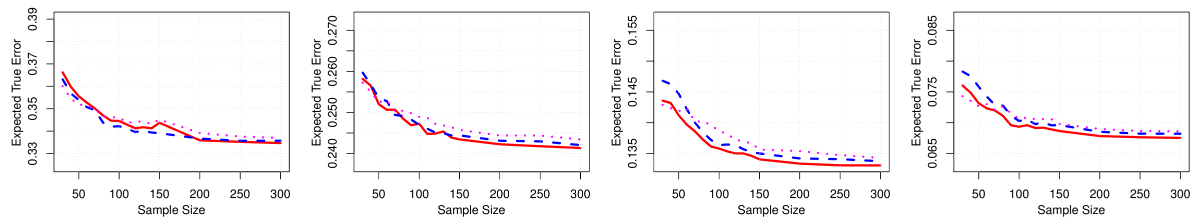
(a) p = 5



(b) p = 20

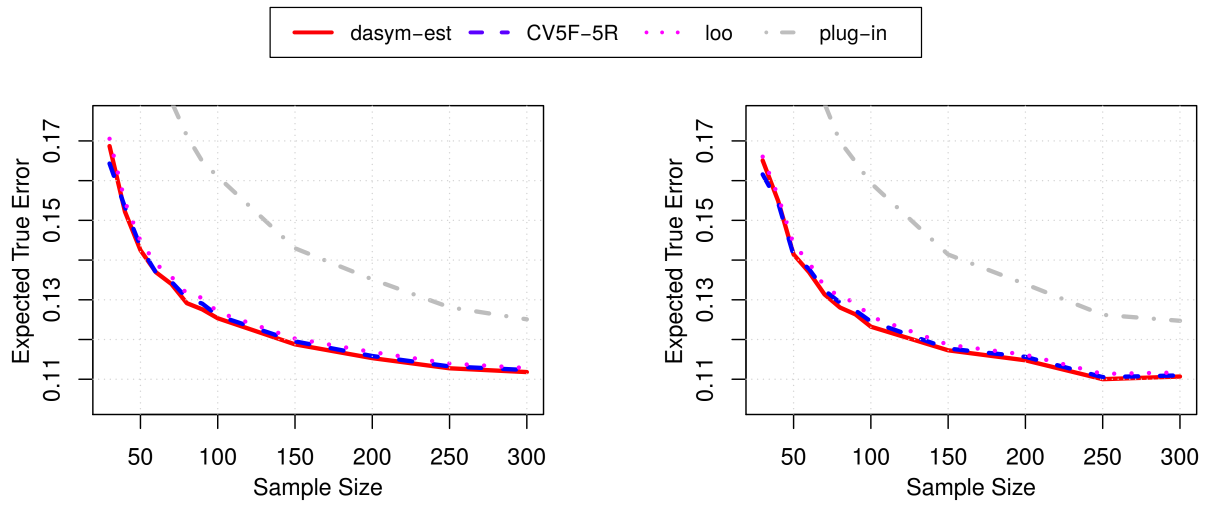


(c) p = 50

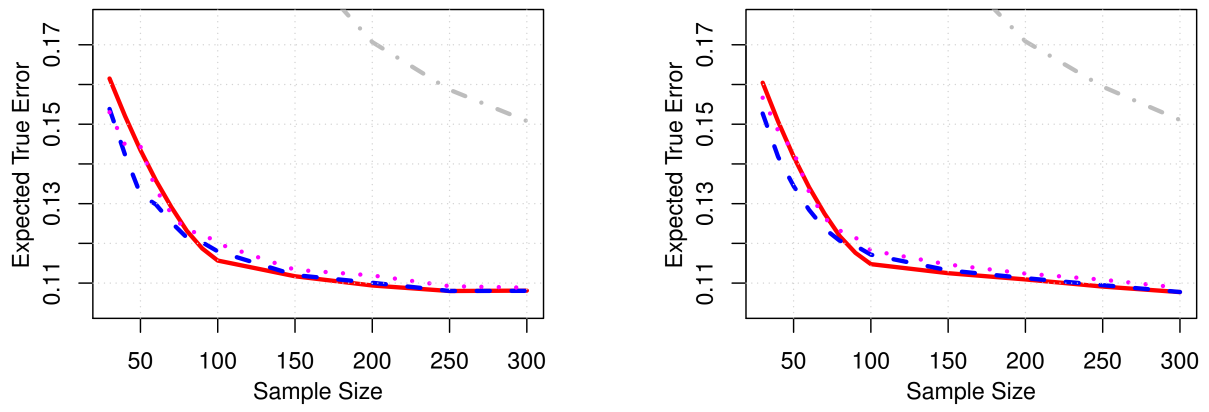


(d) p = 150

Figure 2.6: Expected true error as a function of sample size obtained using synthetic data: case of Gaussian distributions



(a) p = 20



(b) p=50

Figure 2.7: Expected true error as a function of sample size obtained using synthetic data: case of skewed-normal distributions

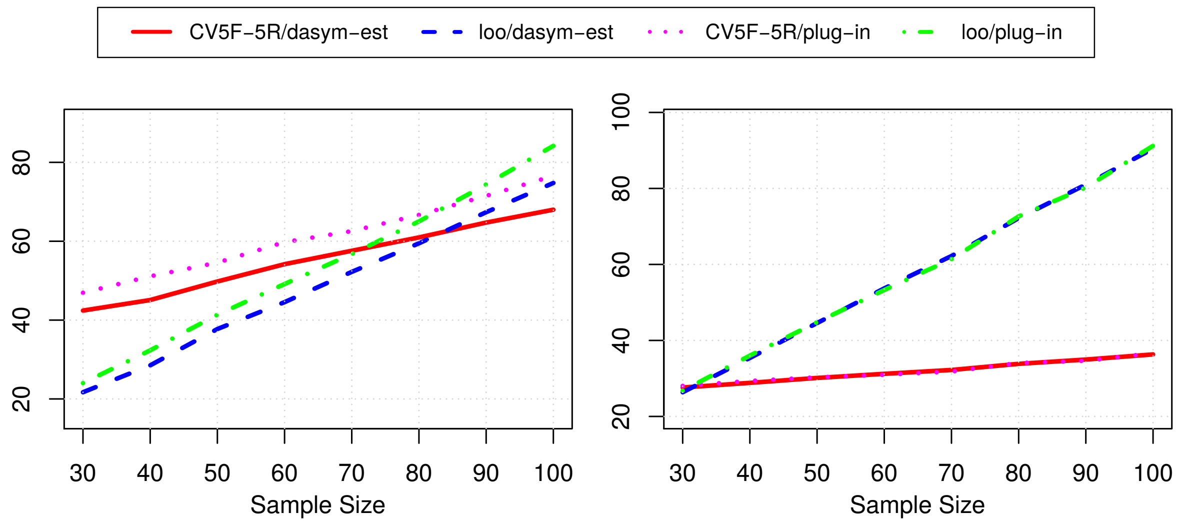


Figure 2.8: Ratio of time spent on different estimators

Chapter 3 – A Comparison of Linear Classifiers when the Sample Size is Comparable to the Dimensionality of Observations

3.1 Introduction

Classification is one of the main features that is offered by machine learning community. Over the last century scientists were attracted by the classical statistical notion of statistical consistency which assumes that the number of observation points are infinitely large, while the dimensionality of observations is fixed ($n \to \infty$, p is fixed). Even though in many modern applications observations possess a dimension comparable to the sample size, many practitioners still attempt to apply the classical techniques.

A simple example of Euclidean Distance Classifier (EDC) given in Chapter 1 and [1] demonstrates the possible potential of high-dimensional machine learning. This example shows depending on the probability structure of observations, there exists classifiers operating in a high dimensional settings that can have better performance than when being used in lower dimension. One potential machinery to construct a classifier well suited for high-dimensional analysis is to use Girko analysis (G-analysis) which is based on the double asymptotic assumptions [1]. The working principle of this analysis is based on having a sample size that is comprable to the magnitude of dimesnion in an asymptotic sense, i.e., $n \to \infty$, $p \to \infty$, $p/n \to c$, $0 < c < \infty$. Since the constant c in $p/n \to c$ can be any finite positive number, the classifiers designed under the double-asymptotic assumptions can potentially operate well in a wide range of sample size and dimension. Wymen et.al [42] conducted a comparison of the analytical expressions designed under classical and double asymptotic assumptions and concluded that the double asymptotic expressions are more accurate than the classical ones, even when $n/p < 3$. Another comparison study was conducted by Raudys and Young [8], who reviewed the analytical expressions of various linear and quadratic discriminant analysis derived under the double asymptotic assumptions. The conclusion of their work reiterates the so called "scissors effect", which means that for a small training sample size, it is more beneficial to apply simple classifiers than complex ones.

A comparative study conducted by Dudoit et.al [14] made a similar conclusion. They compared LDA, Diagonal Linear and Quadratic Discriminant Analysis (DLDA and DQDA), k-nearest neighbor classifiers, classification and regression trees (CART), and aggregating classifiers such as bagging and boosting. They conducted the study on several datasets and set the training sample size as one-third for test and two-third for training, data dimension was picked between $p = 10$ and $p = 200$. This study did not include Support Vector Machines (SVM) and several other classifiers developed based on double asymptotics even though comparable p and n conditions were applied for the experiments. They conclude that the simpler the classifier, the better its performance in small-sample settings. The question that remains is whether constructing a classifier in situations where the sample size and dimension are comparable in magnitude is practically important.

Consider a typical genomic datasets where thousands of genes are studied from a limited number of subjects across various phenotypic groups and the task is to classify individuals based on expression profiles. Regardless of the machinery used, it is now the general consensus that complex traits are commonly characterised by an interplay of many genomic factors. Several studies have cataloged the implicated list of genes in various complex traits and the result show that the number of implicated factors is more or less in the same order of magnitude as a typical genomic dataset, e.g., 500 genes reflecting human melanoma [43] and cervical cancer [44], 240 genes responsible for renal cancer, and more than 100 genes accounting for prostate cancer [45]. The classical statistical approach cannot handle this problem appropriately, while double asymptotic assumptions can potentially lead to an appearance of better analyzing tools.

In this Chapter, we conduct a comprehensive study of several linear classifiers on a set of sample size and dimension that are comparable in magnitude. We consider linear classifiers because they are simple, and consider binary classification as it is very practical. We not only consider popular classifiers such as SVMs, LDA, and RLDA, but also few other relatively unknown linear classifiers developed under a double asymptotic framework.

3.2 Methodology

This section provides a brief overview of nine classifiers used in this chapter. The comparison will be conducted on a samples size varying from 30 to 100 and data dimensionality ranging from 5 to 200. The list includes LDA [23], DLDA [14], EDC [1,14], RLDA [23], G13 [9], Serdobolskii [46], Zarutskij [47] classifiers, linear and non-linear (Gaussian) support vector machines. For all classifiers consider the following points unless specified otherwise: firstly, a binary classification problem is sampled separately from multivariate Gaussian distribution $\mathbb{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$ for $i = 0, 1$. secondly) the sample spaces S_0 and S_1 with n_0 and n_1 ($n = n_0 + n_1$) amount observation vectors are driven from $\mathbb{R}^p$ populations Π_0 and P_1 that have the covariance matrix $\boldsymbol{\Sigma}$ in common. The true error of each classifier in the computational simulations are estimated using hold-out estimator

3.2.1 Classifiers considered in this study

Linear Discriminant Analysis (LDA)

The objective of linear discriminant analysis is to maximize the ratio of the between-class to within-class scattering matrices (see also Section 2.2). The LDA was initially proposed in 1936 in a taxonomic applications [24] and has found various applications today. LDA is represented by Anderson's statistic [23] given by,

$$W^{LDA}(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{C}, \mathbf{x}) = \left(\mathbf{x} - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T \mathbf{C}^{-1} (\mathbf{x}_0 - \mathbf{x}_1), \tag{3.1}$$

where $\bar{\mathbf{x}}_i$ is sample mean for class i,

$$\mathbf{C} = \frac{(n_0 - 1)\mathbf{C}_0 + (n_1 - 0)\mathbf{C}_1}{n_0 + n_1 - 2}, \tag{3.2}$$

where

$$\mathbf{C}_i = \frac{1}{n_i - 1} \sum_{\mathbf{x}_l \in S_i} (\mathbf{x}_l - \bar{\mathbf{x}}_i)(\mathbf{x}_l - \bar{\mathbf{x}}_i)^T. \tag{3.3}$$

Using the discriminant in (3.1), the LDA classifier is then given by

$$\psi_n(\mathbf{x}) = \begin{cases} 1, & \text{if } W^{LDA} \leq c \\ 0, & \text{otherwise} \end{cases}, \tag{3.4}$$

where $c = \log \frac{\alpha_1}{\alpha_0}$.

Diagonal LDA (DLDA)

DLDA is a simple modification of LDA, where the non-diagonal elements in the pooled covariance matrix are replaced by zero:

$$C_{i,j}^{\text{DLDA}} = \begin{cases} C_{i,i} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}, \tag{3.5}$$

where $C_{i,j}$ is given by (3.2). The DLDA classifier is obtained from (3.4) by replacing LDA discriminant with DLDA discriminant.

Euclidean Distance Classifier

As the name suggests the Euclidean Distance Classifier (EDC) calculates the Euclidean distance rather than the Mahalanobis distance (see (1.2) and (3.1)) in LDA. Another way to describe the connection between LDA and EDC lies in the sample pooled covariance matrix modification: the pooled covariance matrix in LDA is replaced by the identity matrix. Either description of EDC is represented by,

$$W^{EDC}(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{x}) = \left(\mathbf{x} - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T (\mathbf{x}_0 - \mathbf{x}_1)\,. \tag{3.6}$$

The EDC follows the same classification rule as LDA shown in (3.4).

G$_{13}$ Classifier

G_{13} classifier resembles LDA much stronger than all other modifications present in this chapter. EDC, DLDA, RLDA and Zarutskij classifiers are related to LDA via the modified inverse sample pooled covariance matrix, while G_{13} classifier scales the whole discriminant value by a certain factor

$$W^{G_{13}}(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{C}, \mathbf{x}) = \left(\mathbf{x} - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T \mathbf{C}^{-1} (\mathbf{x}_0 - \mathbf{x}_1) \left(\frac{n_0 + n_1 - 2 - d}{n_0 + n_1 - 2}\right). \tag{3.7}$$

G_{13} classifier is then obtained from (3.4) by replacing the discriminant by (3.7).

Regularized LDA Classifier

Regularized LDA was originally designed by Di Pillo [17], who based his work on ridge regression (Tikhonov regression) and its regularization parameter. The problem of LDA starts when the observation dimensionality becomes comparable to the sample size or even larger. Such cases produce the ill-conditioned inverse pooled covariance matrix of LDA (see Eq. 3.3). The RLDA replaces the ill-conditioned pooled covariance matrix by a new one:

$$H = \left(\mathbf{I}_p + \gamma \mathbf{C}\right)^{-1}, \tag{3.8}$$

which transforms linear discriminant to

$$W^{RLDA}\left(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{C}, \mathbf{x}\right) = \kappa\left(\mathbf{x} - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T \mathbf{H}\left(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\right), \tag{3.9}$$

where $\kappa > 0, \gamma > 0$. The RLDA classifier is obtained from (3.4) by replacing (3.9) as the discriminant. The generalized consistent estimator introduced in [23] gives an expression to estimate the true error of RLDA:

$$\hat{\varepsilon}_i^D = \Phi\left(\frac{(-1)^{i+1} G\left(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}\right) + \frac{(n_0+n_1-2)\hat{\delta}}{n_i} + (-1)^i \frac{c}{\kappa}}{\sqrt{\left(1+\gamma\hat{\delta}\right)^2 D\left(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}, \mathbf{C}\right)}}\right), \tag{3.10}$$

where

$$\hat{\delta} = \frac{\frac{p}{n_0+n_1-2} - \frac{\mathrm{tr}[\mathbf{H}]}{n_0+n_1-2}}{\gamma\left(1 - \frac{p}{n_0+n_1-2} + \frac{\mathrm{tr}[\mathbf{H}]}{n_0+n_1-2}\right)}, \tag{3.11}$$

while

$$G\left(\boldsymbol{\mu}_i, \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}\right) = \left(\boldsymbol{\mu}_i - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T \mathbf{H}\left(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\right) \tag{3.12}$$

and

$$D\left(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{H}, \mathbf{C}\right) = \left(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\right)^T \mathbf{HCH}\left(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\right). \tag{3.13}$$

The optimum γ which results in the lowest expected true error is estimated via a one dimensional range search for γ, and retrieving the one corresponding to the least estimated true error. The total procedure of γ^{opt} estimation and procedure comparison with popular cross-validation estimators is given in [48] and the previous chapter in this thesis.

Serdobolskii Classifier

Even though Serdobolskii classifier is a modification of LDA, it is not based on the idea of replacing the covariance matrix only. In fact, the classifier original proposal did not include the covariance matrix suggesting that Serdobolskii is generally a modified version of EDC. Serdobolskii in [46] suggests to reduce the dimensionality of the data by simply averaging the data in a feature wise manner to obtain k groups with m elements each and apply EDC:

$$\begin{aligned} \bar{a}_{j,i} &= \frac{1}{m} \sum_{s \in \mathbb{R}_j} \mu_{s,i} \\ z_j &= \frac{1}{m} \sum_{s \in \mathbb{R}_j} x_s, \end{aligned} \tag{3.14}$$

where $\boldsymbol{\mu}_i$ is the population (or sample) mean for class i, $\mathbb{R}_j$ denote a set of numbers $\mathbb{R}_j = \{um, um+1, um+2, \ldots, um+m-1\}$ for $u = \{0, 1, 2, \ldots, k-1\}$, and $\bar{\mathbf{a}}_i = \{a_{0,i}, a_{1,i}, a_{2,i}, \ldots, a_{k,i}\}$ and $\mathbf{z} = \{z_0, z_1, z_2, \ldots, z_{k-1}\}$ for classes $i = \{0, 1\}$. The discriminant function in Serdobolskii classifier is given by,

$$W^{\text{SERD}}\left(\bar{\mathbf{a}}_0, \bar{\mathbf{a}}_1, \mathbf{z}, m\right) = m \left(\mathbf{z} - \frac{\bar{\mathbf{a}}_0 + \bar{\mathbf{a}}_1}{2}\right)^T \left(\bar{\mathbf{a}}_0 - \bar{\mathbf{a}}_1\right). \tag{3.15}$$

To see the performance of the Serdobolskii approach by considering the covariance matrix, we have also devised a heuristic a Serdobolskii-based classifier, by applying the dimensionality reduction strategy first and construct the pooled sample covariance matrix as given in (3.2) and (3.3). Prior constructing it, the whole training data should be translated into lower-dimensional space by using the bottom expression of (3.14) or:

$$a_{j,l} = \frac{1}{m} \sum_{s \in \mathbb{R}_j} \mathbf{X}_{s,l}, \tag{3.16}$$

for classes $l = \{0, 1\}$, where $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{n-1}\}$ is training data matrix and $\mathbf{x}_l$ is observation vector. The discriminant function for Serdobolskii LDA classifier is given by:

$$W_C^{SERD}\left(\bar{\mathbf{a}}_0, \bar{\mathbf{a}}_1, \mathbf{z}, \mathbf{C}_{\text{SERD}}, m\right) = m\left(\mathbf{z} - \frac{\bar{\mathbf{a}}_0 + \bar{\mathbf{a}}_1}{2}\right)^T \mathbf{C}_{\text{SERD}}^{-1}\left(\bar{\mathbf{a}}_0 - \bar{\mathbf{a}}_1\right). \tag{3.17}$$

The decision rule for both Serdobolskii classifiers follow the same strategy as LDA given in (3.4).

Zarudskij Classifier

DLDA classifier uses the diagonal elements of LDA covariance matrices dropping all correlation elements among the features, while EDC simply drops all the relations. Zarudskij classifier suggests the use of a covariance matrix that uses the first order tree-type (FOTT) dependence among variables. The discriminant function of Zarudskij classifier is given by,

$$W^{ZAR}\left(\mathbf{x}, \bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \mathbf{\Sigma}_{\text{Tree}}^{-1}\right) = \left(\mathbf{x} - \frac{\bar{\mathbf{x}}_0 + \bar{\mathbf{x}}_1}{2}\right)^T \mathbf{\Sigma}_{\text{Tree}}^{-1}\left(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\right), \tag{3.18}$$

where $\mathbf{\Sigma}_{\text{Tree}}^{-1} = \mathbf{C}^T\mathbf{C}$ is a $p \times p$ symmetrical FOTT matrix with $3p - 2$ distinct non-zero elements. The $\mathbf{C} = \{c_{ij}\}$ and is given by

$$c_{ij} = \begin{cases} \left(\hat{\sigma}_{ii}\left(1 - r_{im_i}^2\right)\right)^{-\frac{1}{2}} & \text{if } j = i \\ \dfrac{-r_{im_i}}{\sqrt{\sigma_{m_i m_i}\left(1 - r_{im_i}^2\right)}} & \text{if } j = m_i \\ 0 & \text{if otherwise,} \end{cases} \tag{3.19}$$

where $r_{ij} = \dfrac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$ and $\mathbf{m} = \{1, m_1, m_2, m_3, \ldots, m_{p-1}\}$ ($m_0 = 1$ by definition) is a set of numbers indicating the dependence structure among features. To estimate $\mathbf{m}$, Zarudskij suggested to apply the Kruskal's stepwise algorithm which was used to find minimum-spanning tree [47]. The proposed algorithm, firstly, should compute total correlation matrix, and find the maximum absolute value among all branches (rows in a lower triangle in our case). Set $m_0 = 1$, and estimate m_1 to be the next greatest absolute value in the correlation matrix which was not selected before and does not form a cycle with all previous selected branches. This process should be repeated until p elements in $\mathbf{m}$ are not filled. A numerical example showing how to calculate $\mathbf{\Sigma}_{\text{Tree}}^{-1}$ is given in Appendix C.

Linear SVM

The idea behind support vector machines is to find the optimum separation hyperplane between two classes (see [49] for full explanation). Unlike LDA and its modifications, SVM does not use all observation vectors of the training data, instead only the "representatives" of both classes construct the hyperplane which splits the space into two distinct regions. Assume that the space is linearly separable, $\mathbf{x}_i$ is an observation vector, $y_i = \pm 1$ is a SVM decision, where $i = \{0, 1, \ldots, n-1\}$. Let the discriminant function be

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0, \tag{3.20}$$

with the decision rule

$$g(\mathbf{x}) \begin{cases} > 0 & \text{if } \mathbf{x} \text{ corresponds to class 0, then set } y_i = 1 \\ < 0 & \text{if } \mathbf{x} \text{ corresponds to class 1, then set } y_i = -1, \end{cases} \tag{3.21}$$

where

$$\mathbf{w}^T \mathbf{x} + w_0 = 0, \tag{3.22}$$

indicates the between-class separation hyperplane, and

$$y_i(\mathbf{w}^T \mathbf{x}_i + w_0) = 1, \tag{3.23}$$

show the support vectors if observation vector $\mathbf{x}_i$ from each class is the closest one to the separation plane, i.e., the class “representative”. SVM objective is to maximize the distance between the support vectors, which is given by $2/||\mathbf{w}||$. Maximizing the distance in this case is equivalent to minimizing the $||\mathbf{w}||$ to the constraints given in (3.23):

$$\min_{\mathbf{w}} ||\mathbf{w}||_2 \quad \text{subject to} \quad y_i(\mathbf{w}^T \mathbf{x}_i + w_0) = 1, \tag{3.24}$$

where $||\mathbf{a}||_2$ indicates L_2 norm distance of an arbitrary vector $\mathbf{a}$.

Lagrange formalism can solve the optimization problem, the primal form is given by

$$L_p = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\mathbf{w}^T\mathbf{x} + w_0 \right) - 1 \right), \tag{3.25}$$

for $i = 0, 1, \ldots, n-1$, where $\alpha_i \geq 0$ are Lagrange multipliers. Minimizing $\mathbf{w}^T\mathbf{w}$ is equivalent to minimizing L_P with respect to $\mathbf{w}$ and w_0, and maximizing it with respect to α_i. Differentiating the primal form (3.25) with respect to $\mathbf{w}$ and w_0 and equating them to zero leads to

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i}^{n} \alpha_i y_i \mathbf{x}_i. \tag{3.26}$$

Substitute (3.26) into (3.25) constructs the dual form given by

$$L_D = \sum_{i=0}^{n-1} \alpha_i - \frac{1}{2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j. \tag{3.27}$$

The dual form reformulates the original problem (3.24) to a new one

$$\max(L_D) \quad \text{subject to} \quad \alpha_i \geq 0 \quad \sum_{i=0}^{n-1} \alpha_i y_i = 0, \tag{3.28}$$

with a solution given by

$$\mathbf{w} = \sum_{i=0}^{n-1} \alpha_i y_i \mathbf{x}_i. \tag{3.29}$$

Non-linear SVM

The difference between linear and non-linear SVM is in the one additional step of transforming the original vector to a higher dimension and then solving for the linear SVM, i.e., linearize the non-linear function and then apply the linear

SVM. The Lagrange dual form for non-linear SVM is given by

$$L_p = \sum_{i=0}^{n-1} \alpha_i - \frac{1}{2} \sum_{i=0}^{n-1} \sum_{j}^{n-1} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j), \tag{3.30}$$

where $\boldsymbol{\phi}(.)$ is some linearization function. Non-lienar SVM optimization problem is the same as for linear SVM (3.28)

$$\max(L_D) \quad \text{subject to} \quad \alpha_i \geq 0 \quad \sum_{i}^{n-1} \alpha_i y_i = 0$$

and solution for the problem is given by

$$\mathbf{w} = \sum_{i=0}^{n-1} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i). \tag{3.31}$$

In this project we used the Gaussian linearization function given by

$$K(\mathbf{a}, \mathbf{b}) = \boldsymbol{\phi}(\mathbf{a}^T) \boldsymbol{\phi}(\mathbf{b}) = \exp\left(-\frac{|\mathbf{a} - \mathbf{b}|^2}{\sigma^2} \right), \tag{3.32}$$

for arbitrary vectors **a** and **b**, other linearization functions can be found from p.191 in [49]. The popularity of SVM with kernel functions over many other classifiers is related to its optimal hyperplane estimation feature, which provides a "curse of dimensionality" robustness property (for more information read p.432 in [50]). This means that this type of classifiers are expected to not follow the "curse of dimensionality" phenomenon in high-dimensional settings, but still the question [51] is: "For a specific task what is the optimal kernel function to be deployed and how to select a kernel function (kernel selection guidelines)?"

3.2.2 Experiment Models

The classifiers performance were compared on several real datasets under various set of sample sizes and number of features. The sample size varies from $n = \{30, 40, \ldots, 100\}$, while feature size is $p = \{5, 20, 50, 100, 200\}$; this is equivalent to p/n variation from $20/3$ to 0.05, which covers a wide range of the ratio between sample size and dimension. Eight sample sizes, five feature sizes, and eight datasets we have had in this investigation results in 320 experiments in total. In each setting, we consider all 10 classifiers: LDA, DLDA, EDC, G_{13}, RLDA, Serdobolskii original proposal (identified by "sorg" in figures) and with the covariance matrix addition (identified by "scov"), Zarutskij, Linear SVM, and Gaussian kernel SVM. The experimental procedure is described below (pseudo-code is provided in E:

Protocol 3.1:

Step 1: Let $r = N_0/N_1$ express the ratio of the number of observations in class 0 and class 1, and $N = N_0 + N_1$ be the total number of samples in the data. Select a set of observations of size n ($n < N$) and generate a training set such that the proportion of sample points from either class follows the value of r obtained from the full dataset. Let $n_0 = \lfloor r n_1 \rfloor$, where $\lfloor . \rfloor$ indicates the floor function. Setting $n = n_0 + n_1$ and having n_0 leads to $n_1 = \lfloor \frac{n}{r+1} \rfloor$. Randomly select a set of training data of size n and set aside the rest for testing.

Step 2: Apply all classifiers and determine the true error from the set of held-out samples in Step **1**.

Step 3: Repeat Steps **1**-**2**, 500 times and find the average true error for each classifier.

For the feature selection we used t-test as it was done in Chapter 2. Prior

Table 3.1: Microarray studies used in Chapter 3 experiments

Dataset	Features	n_0/n_1
Bhattacharjee [52]	$12,600$	$139/64$
Chen [34]	$10,237$	$75/82$
Desmedt [35]	$22,215$	$98/77$
Natsoulis [36]	$8,491$	$120/61$
Rosenwald [37]	$5,013$	$114/89$
Valk [38]	$22,215$	$116/157$
Vijver [39]	$5,003$	$180/115$
Yeoh [40]	$5,077$	$149/99$

to applying the t-test analysis, data has been standardized according to,

$$\mathbf{z}_i = \frac{\mathbf{x}_i - \bar{\mathbf{x}}_i}{V\left(\mathbf{x}_i\right)/\sqrt{N-1}}, \tag{3.33}$$

where $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{N-1}\}$ is original data, and $\mathbf{Z} = \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{N-1}\}$ is the new data, N is the data sample size, and $V(\mathbf{x})$ is the standard deviation of $\mathbf{x}$.

All set of experiments in this Chapter were conducted on datasets described in Table 3.1. All databases were collected from [31], [32] and [33]. Description for each database is given in Appendix A.

3.3 Results and Discussion

Fig. 3.1 and 3.2 show the expected true error of each classifier at different sample and feature sizes and for different datasets: Bhattacharjee et al. [52], Chen et al. [34], Desmedt et al. [35] and Natsoulis et al. [36], Rosenwald et

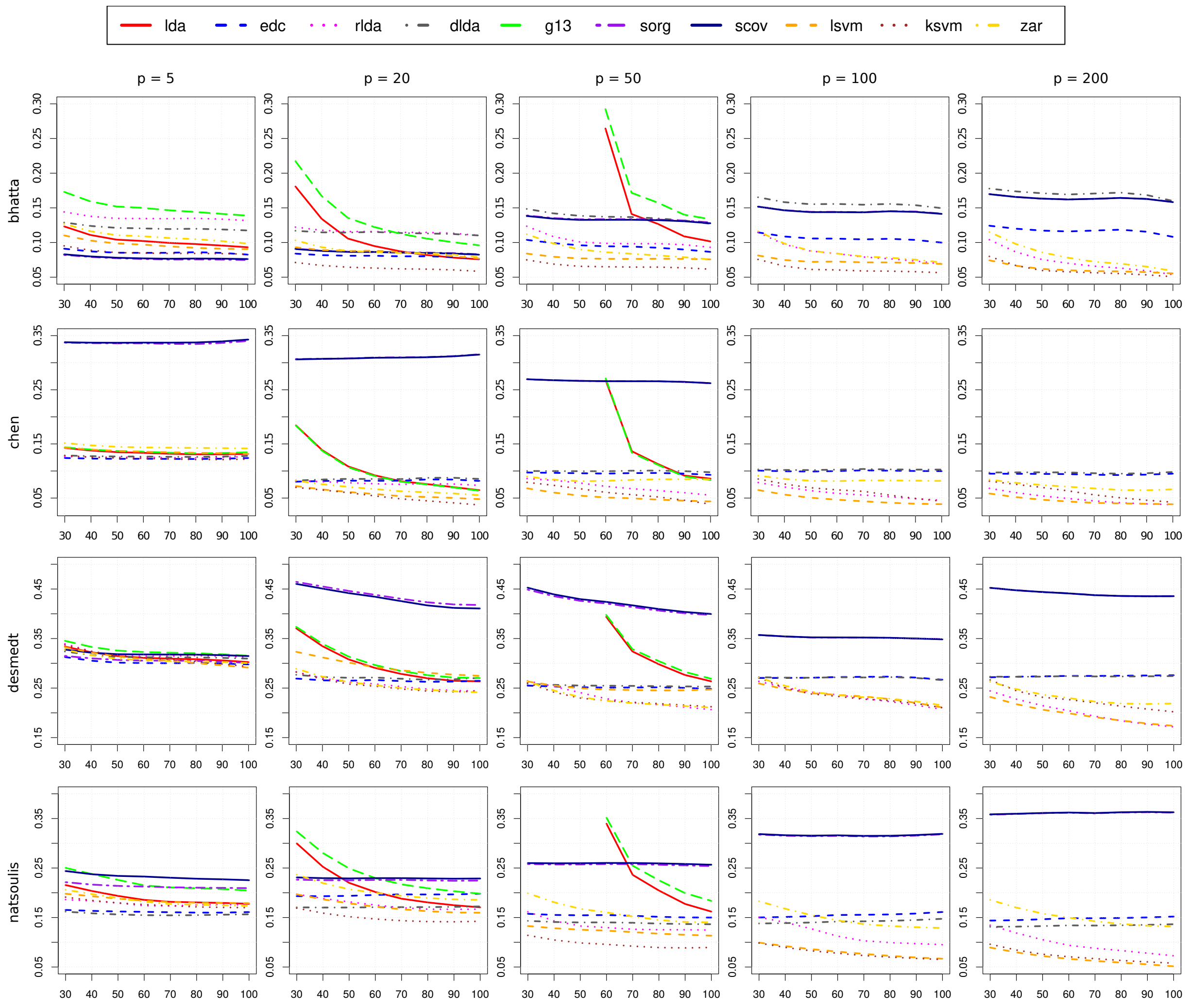


Figure 3.1: Estimated true error of classifiers at different sample sizes

al. [37], Valk et al. [38], Vijver et al. [39] and Yeoh et al. [40] datasets. Each plot has five columns showing the classifier performance at a particular dimension: from left to right $p = 5$, $p = 20$, $p = 50$, $p = 100$ and $p = 200$. Each row in these figures depicts the results for a particular dataset. Nevertheless, in order to simplify the comparison between different classifiers, we have attempted to rank classifiers based on the number of times they outperform others across all experiments (Tables 3.2-3.6). Ranking the classifier performance at each sample and feature size on a particular dataset gives an opportunity to check the superiority of the classifier with respect to others. The full set of tables can be found in the Appendix B.

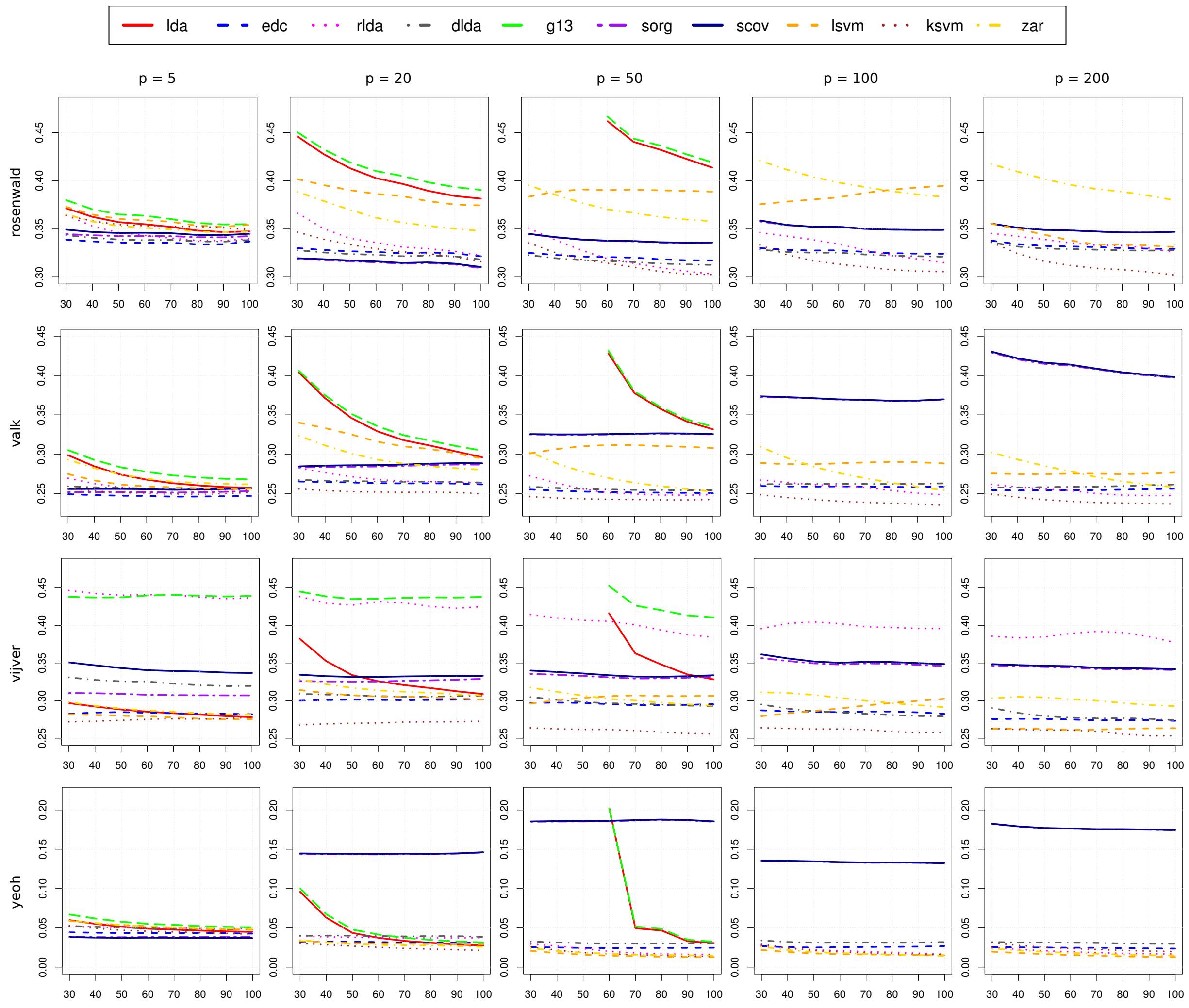


Figure 3.2: Expected true error of classifiers at different sample and feature sizes across different datasetss

One can clearly see that Serdobolskii (sorg and scov) classifiers have non-uniform performance across all feature sizes (for Chen dataset they were omitted). It is unclear how the averaging of features should be accomplished to reflect the optimal classification rates since no feature ordering or ranking was defined by Serdobolskii in [46]. As expected LDA and G13 have relatively good performance when $n >> p$ is met (see left column of Fig. 3.1 and 3.2), however, as soon as data dimension rises, LDA and G13 deteriorates drastically, and their results are omitted from the graphs for $p = 100$, $p = 200$ and partly from $p = 50$ plot. Note that all tables still show the relatively poor performance of these two

Table 3.2: Median value of classifier rank across databases when $p = 5$

Classifiers	Sample Size							
	30	40	50	60	70	80	90	100
LDA	7	7	6.5	6	6	6	6	6
DLDA	4.5	4.5	4.5	6	6	6	7	7
EDC	1.5	1.5	1.5	1.5	2	2	2.5	2.5
G13	10	10	10	10	10	10	10	10
RLDA	6	6	6	6	6	6	6.5	6.5
Serd	2.5	2.5	3	3	4	4.5	4.5	4.5
Serdc	4.5	4.5	6	6.5	6.5	6.5	6.5	7
LSVM	6	6	6	6	6.5	6.5	6	5.5
KSVM	4	4	4	3.5	3.5	3	2.5	2.5
Zar	7	6.5	6.5	6.5	6.5	6.5	6.5	6.5

classifiers with respect to others.

DLDA and Euclidean Distance Classifiers have very similar performances. For all datasets (except Bhattacharjee dataset), the difference between their expected true error rates does not exceed 5%. A close look at the true error rate tendency as the feature size increases might reveal the presence of the "curse of dimensionality" phenomenon at Natsoulis and Vijver datasets. It was shown by EDC example in Chapter 1 and [1] that further increase in dimension might actually result in a better performance, i.e., the true error rate curve beyond the minimum point does not monotonously increase, it can have several local minima, and in our case its global minima might lie outside of the testing feature

Table 3.3: Median value of classifier rank across databases when $p = 20$

Classifiers	Sample Size							
	30	40	50	60	70	80	90	100
LDA	8.5	8.5	7.5	7	6	5.5	5	4.5
DLDA	3	3.5	4.5	4.5	4.5	4.5	5	6
EDC	2	3	4	4	4	4.5	4.5	5
G13	10	10	10	10	8	8	8	8
RLDA	5.5	4.5	4.5	5	5	6	6	6
Serd	5.5	6.5	7	7	8	8	8	8
Serdc	7	7.5	8	8.5	8.5	8.5	8.5	8.5
LSVM	4.5	4.5	4.5	4.5	4.5	3.5	4	4
KSVM	1	1	1	1	1	1	1	1
Zar	6	5.5	5	4	4	5	5	5

size range. Expected true error rate on other datasets show a uniformly increasing function as feature size increases, probably their misclassification rates reached the local minima and now the curve values rise to hit the local maxima in order to go down again. It is interesting to note that expected true error rates of DLDA and EDC do not always decrease as sample size increases, in some cases, as the sample size increases the classifiers performance deteriorates.

Zarudskij classifier uses the FOTT to generate inverse sample pooled covariance matrix. The modification allows the classifier to be used with larger dimensionality and sample size without much deterioration such as those observed in G13 and LDA. Depending on dataset p/n settings, the true error

Table 3.4: Median value of classifier rank across databases when $p = 50$

Classifiers	Sample Size							
	30	40	50	60	70	80	90	100
LDA	10	10	9	9	8	7.5	7	7
DLDA	3.5	5	5	5	5	5	5	5.5
EDC	3	3.5	4.5	4.5	4.5	4.5	4.5	4.5
G13	9	9	10	10	9	9	9	9
RLDA	5	4	4	3	3	3	3	3
Serd	7	7	7	7	8	8.5	8.5	9
Serdc	8	8	7.5	8	9	9	8.5	9
LSVM	2	2.5	3	3	3	3	3	3.5
KSVM	1.5	1	1	1	1	1	1	1
Zar	5	4.5	4.5	4	4	4	4	3.5

of Zarudskij classifier might outperform DLDA and EDC, for larger p and n, Zarudskij is considered to be a better choice than EDC and DLDA.

The objective of RLDA is to eliminate the ill-conditioned inverse covariance matrix from LDA by applying stabilization process (regularization). This results in a classifier that has a good performance in wide range of dimensionality. In larger dimensional feature space, RLDA is in many cases superior to Zarudskij, Serdobolskii, DLDA and EDC. Ironically, the drawback of RLDA is in the regularization parameter, since the γ search boundaries are selected heuristically. To be sure that the RLDA performs well, the classifier should be tested at several γ values around presumed γ^{opt} to see that the estimation

Table 3.5: Median value of classifier rank across databases when $p = 100$

Classifiers	Sample Size							
	30	40	50	60	70	80	90	100
LDA	9	10	9.5	9	9	9	9.5	10
DLDA	5	5	4.5	4.5	5	5	5	5
EDC	3	5	5	5	5	5	5	5
G13	9.5	9	9	10	10	10	9	9
RLDA	4	4	3	3.5	3	3	2.5	2.5
Serd	7	7	7	7	7	7	7	7
Serdc	8	8	8	8	8	8	7.5	7.5
LSVM	2	2	2	2.5	3	2.5	2.5	3
KSVM	2	1	1	1	1	1	1	1
Zar	5	4.5	4.5	4.5	4	4	4	4

procedure has found the local minimum value. This might be the case for the Vijver database, since the RLDA performance significantly differs from all other results.

The best performance in most of the simulations was achieved by KSVM (see Tables 3.2-3.6 and Tables in Appendix B). Although LSVM also shows a relatively good performance, but the underlying assumption in LSVM is that sample space is linearly separable, which might not be case for the particular dataset. Constructing an optimal hyperplane to separate the classes embeds [50] a robustness to the KSVM which results in the best performance achieved in the our simulations.

Table 3.6: Median value of classifier rank across databases when $p = 200$

Classifiers	Sample Size							
	30	40	50	60	70	80	90	100
LDA	10	10	10	10	10	10	9	9
DLDA	5	4.5	4.5	4.5	4.5	4.5	5	5
EDC	4.5	4.5	4.5	5	5	5	5	5
G13	9	9	9	9	9	9	10	10
RLDA	3.5	3	3	3	3	3	3	2
Serd	7	7	7	7	7	7	7	7
Serdc	7.5	7.5	8	7.5	7.5	8	7.5	7.5
LSVM	1	1	1.5	1.5	1.5	2	2	2
KSVM	2	2	1.5	1.5	1.5	1.5	1.5	1.5
Zar	4.5	4.5	4.5	4.5	4.5	4.5	4	4

It was expected that Serdobolskii, Zarutskij and G13 classifiers would show interesting and possibly good performance in our experiments. However, results show that G13 classifier simply follows LDA in terms of expected true error values. Serdobolskii classifier is not fully applicable for classification problems since no strategy is given on how feature averaging should be accomplished, i.e., whether the ranking or any other feature extraction procedure must be applied before taking the averge among feature set. The Zarutskij classifier was designed to eliminate the ill-conditioned inverse covariance matrix, and the results have shown that for high-dimensional problems Zarutskij classifier is a good choice to try.

3.4 Conclusions

Exploring the performance of high-dimensional classifiers is an important topic since in many modern applications we are facing problems where number of observations (samples) is extremely lower that the data dimension. In this Chapter we conduced a comparative study between several popular as well as some relatively unknown linear classifiers. The results show that in many cases, linear and Gaussian kernel SVMs are superior to other classifiers, and the robustness of KSVM to high ratios of p/n allows to avoid the curse of dimensionality problems. The usage of G13 and Serdobolskii classifiers are in question because G13 performs almost similarly to the LDA and they (LDA and G13) both are inapplicable when dimension is comparable to sample size. Although Serdobolskii can have many potential applications, but the averaging procedure used should be reconsidered to reflect the optimal discriminative power of the classifier. Zarutskij classifier shows promising results as it does not have the ill-conditioned covariance matrix and performs well compared to many other classifiers. The RLDA classifier demonstrated good results and as expected it can operate well in high dimensional situations.

Chapter 4 – Sparsity Based Classifiers

4.1 Introduction

High-dimensional nature of modern databases is a challenging task for many classifiers which were designed under classical statistical assumptions, i.e., the number of samples being much higher than the dimensionality of observations. In a finite sample regime this means that in order to expect an acceptable performance from a classical method, the sample size should be much larger than the dimensionality of observations. This discrepancy between the nature of modern datasets and the underlying working principle of many classical methods have led practitioners to reduce the potential dimensionality of the data in the first stage of an analysis to be able to use classical methods. However, this practice may potentially ignore many important features that contribute to the response variable.

Another machinery for analyzing high-dimensional data is based on the shrinkage idea and the sparsity assumption. In this way, it is assumed that the predictive model is sparse in the sense that only a certain number of features contribute to the response variable, while others can be nullified. This approach might resemble the combinatorial analysis which is infeasible in our case and is generally a NP hard problem [53, 54]. Let us momentarily assume a genomic dataset with 5000 genes (features). Suppose one is to find a set of five genes that can reflect the lowest possible error by using a particular classifier. There are

$$\binom{5000}{5} \approx 2.599 \times 10^{16}$$

number of different combinations of five genes (order does not matter). Trying to identify the optimal set of of features by an explicit feature selection strategy is not simply feasible because of the computational time it takes to consider all these configurations of variables. Although many sub-optimal feature selection strategies have been proposed in the past century, shrinkage in conjunction with sparsity assumption provides an alternative approach for feature selection. In this way, the feature selection process is embedded in the model construction stage, hence, performing both processes at the same time.

In 2004 Candes, Tao, Romberg and Donoho [55] discovered that if the signal is sparse, it can be fully restored when the sampling frequency is below the minimum value that is dictated by Shannon theorem. Even though, under-sampling methodologies were already discovered in 1970's in connection with some seismic applications (mineral searching), it was Candes, Tao, Romberg and Donoho who proved that the signal can be restored perfectly with underlying assumptions of signal sparsity.

L_1 minimization is also the main working principle used in the elastic net and lasso regressions. In compressed sensing, a sparse enough signal can be recovered exactly and under certain assumptions the L_1 minimization is the same as L_0 (count of non zero elements) [55]. Basis pursuit and elastic net regression uses L_1 minimization to solve the optimization problems by applying similar approach but on different objective functions, basis pursuit standing [56] for

$$\min_{\mathbf{x}} ||\mathbf{x}||_1 \quad \text{s.t.} \quad \mathbf{Ax} = \mathbf{y} \tag{4.1}$$

and elastic net [57, 58]

$$\min_{(\beta_0, \boldsymbol{\beta})} ||\mathbf{y} - \beta_0 - \mathbf{A}\boldsymbol{\beta}||_2^2 + \lambda \left(\frac{1-\alpha}{2} ||\boldsymbol{\beta}||_2^2 + \alpha ||\boldsymbol{\beta}||_1 \right). \tag{4.2}$$

Note that when $\alpha = 1$, elastic net is the same as lasso, while $\alpha = 0$ resembles the ridge regression.

Both the basis pursuit optimization and lasso regression will be used in this chapter in a classification setting applied in high-dimensional scenarios. The basis pursuit optimization is used in sparse representation classifier (SRC) [59], which was originally designed for face recognition problems. The lasso regression will be converted to a classifier through the logistic regression function [58]. The lasso and logistic regressions methodologies are implemented via glmnet package in R, thus this classifier will be referred as glmnet (binomial glmnet). To successfully apply SRC on image databases, authors in [59] suggested the use of feature extraction methodologies to reduce the data dimensionality. In addition, they propose the “randomfaces” extraction technique which simply generates Gaussian distribution matrix and utilizes the compressed sensing principles. The objective of this section is not to study the feature extraction techniques and their performance on face databases, but to compare different classification rules conducted on high-throughput genomic datasets using SRC and glmnet, where SRC performs the dimensionality reduction via compressed sensing principles.

4.2 Methodology

This section describes the theory behind the Sparse Representation (see original paper [59]) and lasso-based classification (see original papers [57, 58]). A general (not to be confused with generalized) linear model represents an arbitrary response variable y as a sum of arbitrary feature vector $\mathbf{x}$ and $\boldsymbol{\beta}$:

$$y = \mathbf{x}^T \boldsymbol{\beta}, \tag{4.3}$$

where y can be treated as a representation of $\mathbf{x}$ and $\boldsymbol{\beta}$ is treated as weights for each element in the vector $\mathbf{x}$. Extending the linear model to a prediction problem results in the regression analysis, where $\hat{y}$ is a predicted value of the input $\mathbf{x}$ and $\boldsymbol{\beta}$ are the regression coefficients. This means that the linear regression model given in (4.3) attempts to predict y given $\mathbf{x}$ and $\boldsymbol{\beta}$ should be estimated before the prediction, thus there is a training dataset, which trains $\boldsymbol{\beta}$ and a test data, where $\hat{y}$ is predicted.

4.2.1 Binomial glmnet

Consider a binary classification problem, where $\mathbf{y}$ is the sample label and $\mathbf{X}$ is a $[p \times n]$ matrix, where p is the data dimensionality and n is the number of observations. Samples for each class, n_0 and n_1 for class 0 and 1 respectively, are driven separately from $\mathbb{R}^p$ distribution. Let $\boldsymbol{\beta}$ represent the vector of optimal features represent, which has near-zero values for unnecessary features, and non-zero values for selected variables, and let $\mathbf{y}$ show the vector classified testing samples. The elastic net, lasso and ridge regressions estimate the possible outcome value y_i of input $\mathbf{x}_i$, which cannot predict data label. Predicting data outcome label is the primary task of the logistic regression given by:

$$\mathbf{y} = \frac{1}{1 + e^{-(\beta_0 + \mathbf{X}\boldsymbol{\beta})}}, \tag{4.4}$$

where $\boldsymbol{\beta}$ are the regression coefficients. The combination of the logistic and linear regressions gives us the binomial generalized (not to be confused with general) linear model, which can be extended to multiple classes and elastic net regression as well. For classification task we deploy the binomial Generalized Linear Model of elastic Net regression (binomial glmnet) which is given by

$$\min_{(\beta_0, \boldsymbol{\beta})} - \left[\frac{1}{N} \mathbf{y} \left(\beta_0 + \mathbf{X}\beta\right) - \log \left(1 + e^{\beta_0 + \mathbf{X}\boldsymbol{\beta}}\right) \right] + \lambda \left[\frac{1-\alpha}{2} ||\beta||_2^2 + \alpha ||\boldsymbol{\beta}||_1 \right], \tag{4.5}$$

where selecting α to be extreme values 0 or 1 results in ridge and lasso regressions, respectively. The glmnet package provided in R quadratically approximates the log-likelihood and then applies coordinate descent to estimate the $\boldsymbol{\beta}$. This section does not provide the full methodology on coordinate descent algorithm and readers may refer to [58] for more information. Throughout this chapter, we set $\alpha = 1$ to see the difference between basis pursuit (left) and the lasso (right):

$$\min_{\mathbf{x}} ||\mathbf{x}||_1 \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{y} \qquad\qquad \min_{(\beta_0, \boldsymbol{\beta})} ||\mathbf{y} - \beta_0 - \mathbf{A}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1.$$

4.2.2 Sparse Representation Classifier

In 2009 Wright et al. [59] proposed a sparse representation classifier (SRC), which is based on sparsity of the training data sample vector and linear regression given in (4.3). Consider a multiclass problem with K classes, n total sample size, n_i is the number of sample points for class i, and $\mathbf{A}$ is a $[p \times n]$ matrix with p features. In (4.3), $\boldsymbol{\beta}$ term is replaced by $\mathbf{x}$ for convenience and ideally $\mathbf{x}$ should be a $[n \times 1]$ vector containing ones and zeros only, $\mathbf{y}$ is a $[p \times 1]$ testing sample vector. The linear regression equation for SRC with prescribed definitions is given by

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \epsilon. \tag{4.6}$$

where ϵ describes the tolerance to errors and for simplicity it can be set to zero. Since $\mathbf{A}$ is a matrix representing the training samples across all classes ($p \times n$ matrix) and $\mathbf{y}$ is a testing sample, the $\mathbf{x}$ can be considered as a vector which decides what samples in $\mathbf{A}$ are contributing the testing sample, and ideally *ones* should be at those entries which indicate the affiliation (class) of $\mathbf{y}$ and zeros anywhere else. The SRC statement is

$$\min_{\mathbf{x}} ||\mathbf{x}||_1 \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{y} \tag{4.7}$$

or

$$\min_{\mathbf{x}} ||\mathbf{x}||_1 \quad \text{s.t.} \quad ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2 \leq \epsilon, \tag{4.8}$$

which is the same as basis pursuit given in (4.1). The former expression assumes no error is present in $\mathbf{A}$ and $\mathbf{y}$, while the latter one is a more realistic as it accounts for the error in data and tries to minimize the error power. One can use any algorithm which solves (4.8). In this thesis, we use a package called $l_1$magic[1], which was originally provided by Dr. Candes and written in MatLab (although we partially imported the package into R). Since it is likely that the approximate solution $\hat{\mathbf{x}}$ obtained by L_1 contains non-zero entries representing several classes, the classifier should select the class shown in $\hat{\mathbf{x}}$ that is dominant. To do that, the L_2 difference between the class i representatives in $\mathbf{A}$ and testing sample $\mathbf{y}$ is calculated:

$$r_i(\mathbf{y}) = ||\mathbf{y} - \mathbf{A}\boldsymbol{\delta}_i(\hat{\mathbf{x}})||_2, \tag{4.9}$$

for $i = \{0, 1, \ldots, K-1\}$, where $r_i(\mathbf{y})$ is the residual function of test sample $\mathbf{y}$ at class i, $\boldsymbol{\delta}_i(\hat{\mathbf{a}})$ is an indicator function, which has non-zero values in $\hat{\mathbf{a}}$ at entries that correspond to samples of class i. The estimated class label is the one that results in the minimum residual value. The SRC methodology can be summarized as following:

- Construct matrix A and normalize its column entries to a unit vector
- Solve L_1 normalization problem given by

  $$\min_{\mathbf{x}} ||\mathbf{x}||_1 \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{y}$$

  or

  $$\min_{\mathbf{x}} ||\mathbf{x}||_1 \quad \text{s.t.} \quad ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2 \leq \epsilon,$$

- Find the residual values according to

  $$r_i(\mathbf{y}) = ||\mathbf{y} - \mathbf{A}\boldsymbol{\delta}_i(\hat{\mathbf{x}})||_2,$$

- Select expected class to be $\arg\min_i (r_i)$.

Similar to LDA classifier, SRC cannot operate (basis pursuit solver) in

[1]https://statweb.stanford.edu/ candes/l1magic/

cases when data dimensions is comparable or greater than the number of samples. The original study [59] compared SRC and several other classifiers with different feature extraction techniques such as eigenfaces, laplacianfaces, fisherfaces and so called “randomfaces”, which was proposed by the same authors. The results show that for a particular usage with SRC, the best feature extraction is randomfaces, which does not have the best performance with other examined classifiers such as Nearest Neighbor, Nearest Space, or SVM. This suggests that the sparsity utilization in SRC is well aligned with compressed sensing theory. The randomfaces extraction method is based on a normally distributed matrix that is featurewise normalized to a unit vector which should be multiplied by all samples available in the data.

A good symbiosis between randomfaces and SRC might be the result of the compressed sensing applications and theory. Even though SRC and the compressed sensing have different objectives (classification and signal reconstruction), the overall methodology is very similar. Assume that in (4.6) $\mathbf{y}$ is the data that is sensed with a sampling frequency that is far lower than the one that is dictated by Shannon theorem, $\mathbf{x}$ be an original signal which should be reconstructed, and let $\mathbf{A}$ be a dictionary matrix, i.e., the transformation matrix that converts the readings into the desired representation of the signal. A dictionary matrix might be thought of a domain transformation matrix, for example the transformation from frequency domain to a time domain or from wavelet domain to a spatial (pixel) domain. This resembles the SRC methodology of finding the $\mathbf{x}$ given $\mathbf{A}$ and $\mathbf{y}$, and if the randomly generated dictionary matrix is well suited for the reconstruction problems, then so the randomfaces should be.

Assume that the original signal has K non zero coefficients (K-sparse) out of T total amount of coefficients, where $T >> K$, the amount of measurements performed is M. In order to apply compressed sensing theory [55], the number of measurements should satisfy following condition:

$$M \approx K \log \left(\frac{T}{K} \right), \tag{4.10}$$

which means that for sample size of 50 and 2-sparse (one best representative of each class) $\mathbf{x}$ of SRC (see Eq. 4.7), theoretically it is enough to have 7 coefficients in $\mathbf{y}$ to reconstruct the $\mathbf{x}$ showing that SRC can operate in a very low-dimensional space.

4.3 Systems and Models

Table 4.1: Microarray studies used in Chapter 4 experiments

Dataset	Features	n_0/n_1
Bhattacharjee [52]	$12,600$	$139/64$
Chen [34]	$10,237$	$75/82$
Desmedt [35]	$22,215$	$98/77$
Natsoulis [36]	$8,491$	$120/61$
Rosenwald [37]	$5,013$	$114/89$
Su [60]	$12,553$	$83/91$
Valk [38]	$22,215$	$116/157$
Vijver [39]	$5,003$	$180/115$

Classification rules comparison was conducted on eight real databases, Table 4.1 provides a short summary of each them, for total descriptions one may refer to Appendix A. We are interested in comparison of the lasso regression and basis pursuit optimization problems, thus α is set to 1 in (4.5). Both of the classifiers do feature selection procedures and the optimum amount of feature is estimated via k-fold cross validation techniques, glmnet deploys 10 fold CV, SRC uses 10 fold 10 repetition CV. For SRC the feature selection procedure was selected as the factor of the sample size, while glmnet package estimates the $\boldsymbol{\beta}$ via coordinate descent algorithm and $\boldsymbol{\beta}$ can have any amount of non-zero coefficients unless data sample size is not exceeded. Protocol used for experiments described below (the pseudo-codes are provided in F:

Protocol 4.1:

Step 1: Let $r = N_0/N_1$ express the ratio of total amount of features of class 0 and 1. Let $n_1 = \lfloor n/(r+1) \rfloor$ and $n_0 = n - n_1$, where n is the learning set sample size and known prior the experiment and sample randomly training and testing sets. Let $\alpha_0 = \frac{N_0}{N}$ and $\alpha_1 = \frac{N_1}{N}$, this kind of learning set separation ensures that training set closely follows the parameters of total database.

Step 2: Estimate $\boldsymbol{\beta}$ via glmnet package (cv.glmnet function)

Step 3: Let $\mathbf{f}_v = \{10, 8, 5, 3, 2, 1.5, 1.25\}$, let SRC dimensions to be learned with $\mathbf{p}_v = n/\mathbf{f}_v$, i.e., divide sample size by each value of $\mathbf{f}_v$. Apply CV10F-10R with SRC and estimate optimum dimension.

Step 4: Compute the true error of SRC and binary glmnet using values estimated before.

Step 5: Repeat all previous Steps 500 times. The estimated expected true error is the mean value of all repetitions.

Table 4.2: Dimension Frequency on Bhattacharjee Dataset

Dimension	Binomial glmnet								SRC							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
10	24	5	7	6	3	2	3	0	87	29	43	39	25	38	30	38
8	0	9	4	2	3	3	3	7	0	62	53	56	50	45	52	36
5	56	36	48	43	52	53	67	83	88	98	106	103	99	129	117	111
3	127	123	141	182	191	203	215	225	112	133	117	142	167	153	138	169
2	174	233	241	236	225	232	205	184	100	85	107	108	117	94	124	121
1.5	112	88	59	31	26	7	7	1	77	68	63	44	35	33	38	22
1.25	7	6	0	0	0	0	0	0	36	25	11	8	7	8	1	3

4.4 Results and Discussion

Fig. 4.1 and 4.2 show the expected true error of binomial glmnet and SRC on all tested datasets. In most of the cases except the “Desmedt" and “Rosenwald" ones (Fig. 4.1c and 4.2a), the glmnet outperforms the SRC regardless of the sample size. The possible reason for glmnet’s poor performance in two datasets might be data themselves, the Desmedt and Rosenwald datasets have a relatively high true error rate (around 45% when $n = 30$), this might imply that for difficult classification it is more beneficial to utilize the SRC rather than the binomial glmnet.

Tables 4.2-4.9 show the frequency of estimated optimal amount of features. The optimal dimension selection procedure for SRC allows us to define the distinct values of dimensions to search through, while the glmnet attempts to

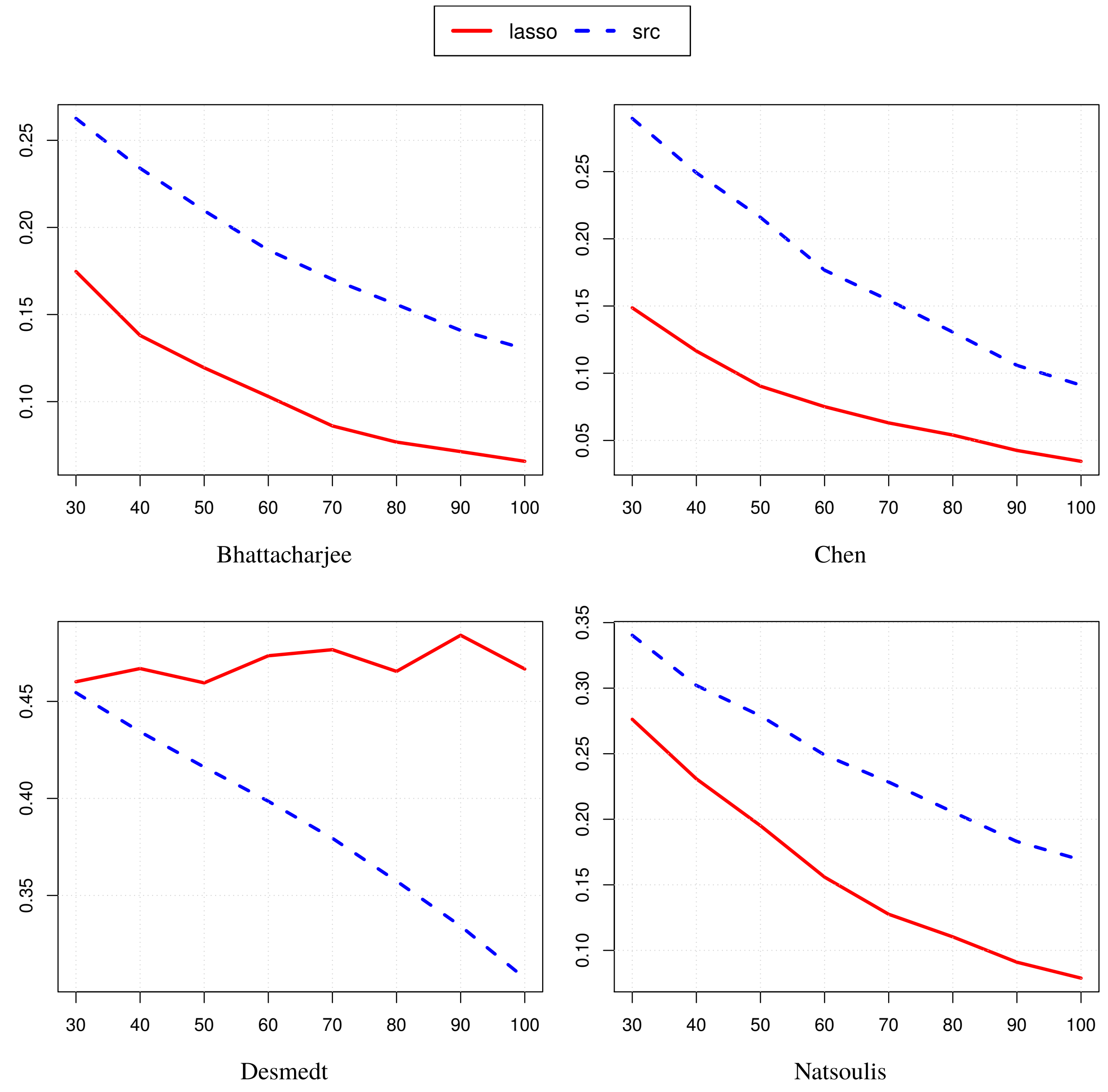


Figure 4.1: Expected estimated and true error (y-axis) at different sample sizes (x-axis)

estimate the optimum $\boldsymbol{\beta}$ meaning that every dimension that is less than the sample size has a chance to be selected. This implies that the tables for binomial glmnet calculate the histogram values, i.e., the number of times when dimensions were selected within a certain range of values, while SRC tables show the counts of distinct dimension value. It is seen that in the case of Desmedt and Rosenwald datasets and for the SRC, none of the possible dimensions is dominant as it is the case for all other databases. For Valk and Vijver datasets the distributions of optimum dimension between glmnet and SRC differ in terms of the the maximum

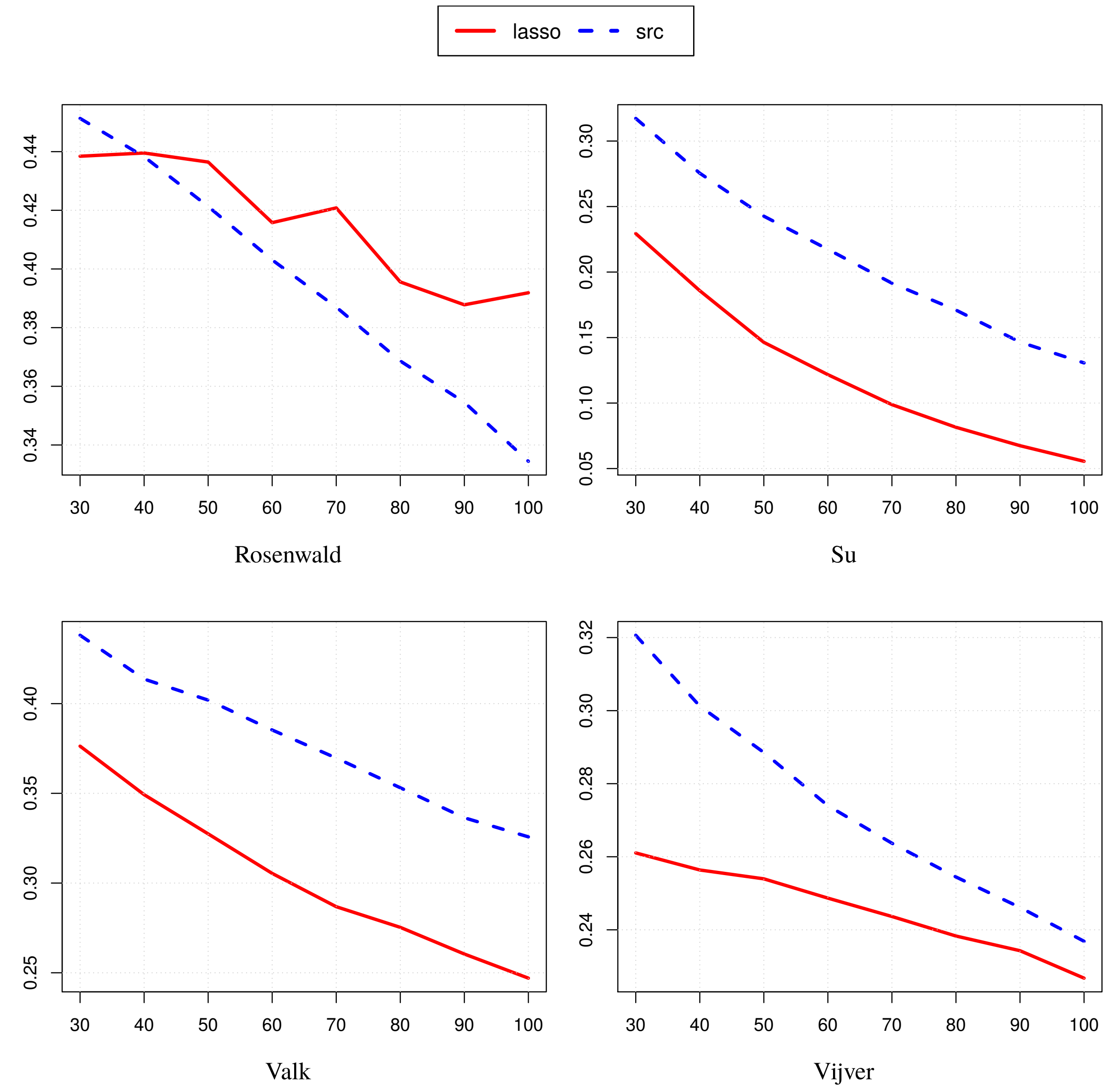


Figure 4.2: Expected estimated and true error (y-axis) at different sample sizes (x-axis)

frequency value, for glmnet the lowest dimensions dominates all others, while for SRC the most frequent optimum dimension is shifted towards $n/2$ values. The same tendency of low optimum dimensions in binomial glmnet is observed for Desmedt and Rosenwald datasets, however the true error values and its behivour versus the sample size between Desmedt-Rosenwald and Valk-Vijver datasets differ significantly.

Another significant difference between two classification rules is related to the peak values of optimum dimension distribution, regardless of the glmnet

Table 4.3: Dimension Frequency on Chen Dataset

Dimension	Binomial glmnet								SRC							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
10	55	39	20	18	19	9	5	2	92	33	41	25	14	10	9	10
8	0	12	9	5	7	11	5	3	0	30	34	29	28	14	17	13
5	53	49	65	46	45	51	42	45	69	67	58	54	51	65	64	58
3	80	89	98	143	133	148	173	188	83	106	116	131	124	126	144	151
2	128	167	192	200	228	248	257	250	106	125	128	136	170	161	172	177
1.5	134	121	110	85	68	33	18	12	83	95	85	98	90	111	79	81
1.25	49	23	6	3	0	0	0	0	67	44	38	27	23	13	15	10

true error performance, the distribution variances for glmnet at all datasets except the Valk's one are much smaller comparing to SRC, i.e., the frequency of dominant dimension selection via glmnet is much higher than the SRC ones. The possible explanation for than might lie in the optimum frequency range selection procedure. There might be the situation when the optimum dimensions that can be calculated by SRC lies on the edge between two consecutive factors predefined in f_v.

4.5 Conclusions

The usage of L_1 minimization algorithms in machine learning is very well justified if data is assumed to be sparse. Although the basis pursuit and elastic net (lasso in our case) expressions deploy the L_1 minimization, they have different objective functions. Based on that, this chapter shows how basis pursuit and lasso

Table 4.4: Dimension Frequency on Desmedt

Dimension	Binomial glmnet								SRC							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
10	265	278	236	286	274	254	276	248	151	77	66	82	77	78	78	76
8	0	10	8	6	5	13	10	8	0	62	85	81	78	70	66	50
5	43	31	43	37	28	33	42	37	76	84	75	60	72	66	61	78
3	40	46	38	37	42	42	38	41	72	65	67	78	62	81	81	88
2	51	48	69	48	50	48	35	45	65	68	67	68	82	65	87	82
1.5	55	33	53	53	45	47	54	45	70	77	73	64	79	73	76	60
1.25	35	44	48	27	47	54	40	69	66	67	67	67	50	67	51	66

are used for classification problems. The lasso (and elastic net as well) construct the binomial generalized linear model of elastic net (binomial glmnet), while the basis pursuit is the foundation of sparse representation classifier (SRC). The binomial glmnet is based on shrinking the data and assuming that only several features can best represent the data response, while SRC assumes that only several samples from training set best represent the testing sample, thus training sample vector is assumed to be sparse. Both of the classifiers cannot operate in cases when data dimension is comparable or greater than the sample size implying that feature extraction procedure should be used. This is not required for binomial glmnet as its objective is to find the optimum dimension, while for SRC the dimension reduction strategy was defined on trial and test manner, and the estimating procedure is based on a k-fold, r-repetition cross validation.

The experiments conducted on eight real datasets show that in general in

Table 4.5: Dimension Frequency on Natsoulis

Dimension	Binomial glmnet								SRC							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
10	97	27	12	6	1	0	0	0	84	28	24	18	8	6	4	3
8	0	7	2	5	0	1	0	0	0	24	27	24	10	11	6	12
5	43	48	30	24	13	7	14	7	65	44	53	34	43	38	35	46
3	109	88	94	88	74	91	96	86	80	102	101	89	105	89	105	106
2	118	163	198	214	274	282	305	361	100	111	112	134	152	166	180	183
1.5	115	146	154	156	138	118	85	46	98	106	122	144	128	140	141	116
1.25	16	21	10	7	0	1	0	0	73	85	61	57	54	50	29	34

simpler classification problems (low true error of classifier), binomial glmnet uniformly outperforms the SRC. The performance of two datasets (Desmedt and Rosenwald) show that when the classification is difficult (relatively high misclassification rate) it is beneficial to use SRC. The dimensionality reduction strategy of SRC mostly points to $n/2$ or $n/3$ as the optimum dimension, where n is the sample size. The optimum dimesnion determined by the binomial glmnet is not uniform across all datasets. In some cases glmnet has optimum dimension frequency distribution very similar to SRC, but for others the distribution mean value is shifted towards lower dimensions. It is certain that the peaks of distribution mean value is much sharper for the glmnet. The possible future work in this field is related to replacing the basis pursuit framework used in SRC by the elastic net or lasso and study their performances.

Table 4.6: Dimension Frequency on Rosenwald

Dimension	Binomial glmnet								SRC							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
10	216	219	207	166	175	139	133	158	159	81	81	92	79	75	80	65
8	0	14	13	12	21	15	17	19	0	86	86	62	67	65	71	78
5	46	39	42	49	57	50	50	49	70	78	78	70	69	83	79	86
3	57	52	53	93	62	73	76	61	64	75	68	72	68	71	70	82
2	56	77	83	63	74	74	78	74	64	69	56	69	85	78	72	58
1.5	71	54	61	82	69	99	109	96	72	61	73	70	69	68	60	80
1.25	48	41	39	31	41	45	37	43	71	50	58	65	63	60	68	51

Table 4.7: Dimension Frequency on Su

Dimension	Binomial glmnet								SRC							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
10	24	23	16	6	11	4	1	1	76	23	23	21	25	22	14	14
8	0	14	1	7	2	7	1	1	0	34	38	35	31	29	32	26
5	42	24	22	21	16	14	11	13	79	75	73	87	72	77	92	84
3	60	75	58	72	73	74	69	67	105	100	115	111	129	125	123	137
2	97	124	139	151	172	168	187	245	100	121	118	128	143	134	156	164
1.5	157	141	201	209	202	220	227	172	76	88	88	89	76	99	75	69
1.25	100	94	61	34	24	13	4	1	64	59	45	29	24	14	8	6

Table 4.8: Dimension Frequency on Valk

Dimension	Binomial glmnet								SRC							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
10	177	152	116	117	96	113	115	108	128	62	55	58	50	43	41	58
8	0	18	9	19	17	26	21	21	0	52	68	62	69	65	56	61
5	44	46	64	53	91	77	76	85	61	75	69	80	79	86	100	85
3	69	68	76	83	93	78	81	93	86	97	85	79	84	88	97	86
2	81	96	98	86	88	97	78	73	74	83	78	93	87	97	89	102
1.5	78	54	78	83	63	71	83	84	79	88	79	87	84	78	81	61
1.25	41	59	56	53	50	36	46	36	72	43	66	41	47	43	36	47

Table 4.9: Dimension Frequency on Vijver

Dimension	Binomial glmnet								SRC							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
10	328	304	258	289	277	256	263	268	132	41	39	48	40	43	33	35
8	0	13	13	11	13	12	9	17	0	53	45	48	57	43	62	32
5	41	47	40	48	42	38	66	48	73	70	63	81	76	72	79	89
3	52	49	59	59	64	85	77	74	94	97	104	120	109	110	106	127
2	51	54	91	64	76	79	55	73	92	119	119	106	112	126	124	120
1.5	26	29	37	29	27	29	30	20	67	78	98	69	82	82	79	71
1.25	2	4	2	0	1	1	0	0	42	42	32	28	24	24	17	26

Chapter 5 – Conclusions

High-dimensionality of observations poses the signal processing communities with great challenges. This is because many classical methods developed in the past are grounded in classical asymptotic conditions ($n \to \infty$, p fixed). In a finite sample regime, this means having many more sample points than the number of variables; however, today we are facing with many datasets in which the number of dimensions is much larger than the sample size. As a result the performance of many classical techniques needs to be re-examined and new techniques need to be developed.

In three chapters, we show that high-dimensionality should not repulse the community from deploying classification techniques directly in a high-dimensional setting. In Chapter 2, we introduce a novel approach to estimate the regularization parameter γ in Regularized Linear Discriminant Analysis (RLDA). Our approach is based on a general consistent estimation which is a mathematical framework designed to create estimators, which converge to the actual parameters in a double-asymptotic regime ($n \to \infty$, $p \to \infty$, $p/n \to c$, $0 < c < \infty$). We compared the performance of the proposed range search of optimum regularization parameter based on our estimator with several other popular schemes such as five folds, five repetitions and leave-one-out cross validation as well as the plug-in estimator. While the performance of constructed RLDA classifiers using the proposed search strategy is similar or better than cross-validation-based search, the analytical expression of the core estimator in our appraoch provides an opportunity to avoid repetitive computations performed by CV. As a result the proposed search scheme is tens to hundreds of times faster to compute.

Chapter 3 conducts a comprehensive comparative analysis between several classifiers designed for low- and high-dimensional settings including LDA, Diagonal LDA, RLDA, Euclidean Distance Classifier (EDC), G13, Serdobolskii, Zarutskij as well as Linear and Gaussian Kernel Support Vector Machines (LSVM and KSVM). Experiments were conducted for cases where the ratio of dimension to sample size varies between 0.05 and 7 (approximately). The results show that the best performance is generally achieved by KSVM and LSVM, while Serdobolskii classifier that was deliberately designed for high-dimensions did not show satisfactory results. Nevertheless, in the case of Serdobolskii classifier further research needs to be conducted to propose an optimal feature-wise averaging scheme. This might lead to significant increase in classification accuracy of this classifier.

Another set of machinery to design classifiers that are applicable in high-dimensional settings is based on L_1 minimization and model sparsity. Chapter 4 compares two classification rules based on this idea, namely, classification based on lasso and basis pursuit. The generalized linear model of lasso regression-based (binomial glmnet) classifier assumes that only certain number of features are required to represent the response variable, thereby shrinking the data dimension. Sparse representation classifier (SRC) is based on basis pursuit and assumes that only a number of samples of a particular class are needed to represent the test data. Results show that in general binomial glmnet is superior to SRC, while for difficult classification problems the performance is vice versa.

To summarize, this thesis provides concrete applications of two high-dimensional mathematical machineries, namely, double asymptotics and L_1

minimization in conjunction with sparsity. Further research needs to be done to: 1) extend the applications of these machineries to Bayesian settings; 2) better clarify the working principle behind the feature selection process implicit in L_1 minimization; and 3) investigate the possibility of integrating these two potential approach together.

Bibliography

[1] A. Zollanvari, “High-dimensional statistical learning: Roots, justifications, and potential machineries,” *Cancer Informatics*, pp. 109–121, 04 2016. [Online]. Available: www.la-press.com/high-dimensional-statistical-learning-roots-justifications-and-potenti-article-a5528

[2] A. Korostelev and O. Korosteleva, *Mathematical Statistics: Asymptotic Minimax Theory*, ser. Graduate studies in mathematics. American Mathematical Soc., 2011.

[3] A. B. Tsybakov, *Introduction to Nonparametric Estimation*, ser. Springer Series in Statistics. Dordrecht: Springer, 2009. [Online]. Available: https://cds.cern.ch/record/1315296

[4] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.

[5] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[6] B. Chandrasekaran and A. K. Jain, “Quantization complexity and independent measurements,” *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 102–106, Jan 1974.

[7] S. J. Raudys and A. K. Jain, “Small sample size effects in statistical pattern recognition: recommendations for practitioners,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–264, Mar 1991.

[8] S. Raudys and D. M. Young, “Results in statistical discriminant analysis: A review of the former soviet union literature,” *Journal of Multivariate Analysis*, vol. 89, pp. 1–35, 2004.

[9] V. Girko, *An Introduction to Statistical Analysis of Random Arrays*. VSP, 1998.

[10] R. Couillet and M. Debbah, *Random Matrix Methods for Wireless Communications:*. Cambridge University Press, Aug 2011. [Online]. Available: https://www.cambridge.org/core/books/random-matrix-methods-for-wireless-communications/6A432BE9C2FE112D0E3F4B7B6568F4F9

[11] Z. Bai and J. W. Silverstein, *Spectral analysis of large dimensional random matrices*. Springer, 2010.

[12] M. Pourahmadi, *High-Dimensional Covariance Estimation*. Wiley, 2013.

[13] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. [Online]. Available: http://www.jstor.org/stable/2346178

[14] S. Dudoit, J. Fridlyand, and T. P. Speed, "Comparison of discrimination methods for the classification of tumors using gene expression data," *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 77–87, 2002. [Online]. Available: http://www.jstor.org/stable/3085760

[15] P. N. Lawlor, T. Kalisky, R. Rosner, M. R. Rosner, and K. P. Kording, "Conceptualizing cancer drugs as classifiers," *PLOS ONE*, vol. 9, no. 9, pp. 1–10, 09 2014. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0106444

[16] H. Kang, I.-M. Chen, C. S. Wilson, E. J. Bedrick, R. C. Harvey, S. R. Atlas, M. Devidas, C. G. Mullighan, X. Wang, M. Murphy, K. Ar, W. Wharton, M. J. Borowitz, W. P. Bowman, D. Bhojwani, W. L. Carroll, B. M. Camitta, G. H. Reaman, M. A. Smith, J. R. Downing, S. P. Hunger, and C. L. Willman, "Gene expression classifiers for relapse-free survival and minimal residual disease improve risk classification and outcome prediction in pediatric b-precursor acute lymphoblastic leukemia," *Blood*, vol. 115, no. 7, pp. 1394–1405, 2010. [Online]. Available: http://www.bloodjournal.org/content/115/7/1394

[17] P. J. D. Pillo, "The application of bias to discriminant analysis," *Communications in Statistics - Theory and Methods*, vol. 5, pp. 843–854, 1976.

[18] A. E. Hoerl, “Application of ridge analysis to regression problems,” *Chemical Engineering Progress*, vol. 58, pp. 54–59, 1962.

[19] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, pp. 55–59, 1970.

[20] ——, “Ridge regression: Applications to nonorthogonal problems,” *Technometrics*, vol. 12, pp. 69–82, 1970.

[21] P. J. D. Pillo, “Biased discriminant analysis: Evaluation of the optimum probability of misclassification,” *Communications in Statistics - Theory and Methods*, vol. 8, pp. 1447–1457, 1979.

[22] R. Peck and J. V. Ness, “The use of shrinkage estimators in linear discriminant analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 4, pp. 409–424, 1982.

[23] A. Zollanvari and E. R. Dougherty, “Generalized consistent error estimator of linear discriminant analysis,” *IEEE Transactions on Signal Processing*, vol. 63, no. 11, pp. 2804–2814, June 2015.

[24] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. [Online]. Available: http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x

[25] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. fisherfaces: recognition using class specific linear projection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, Jul 1997.

[26] S. Kim, E. R. Dougherty, I. Shmulevich, K. R. Hess, S. R. Hamilton, J. M. Trent, G. N. Fuller, and W. Zhang, “Identification of combination gene sets for glioma classification,” *Molecular Cancer Therapeutics*, vol. 1, no. 13, pp. 1229–1236, 2002. [Online]. Available: http://mct.aacrjournals.org/content/1/13/1229

[27] E. I. Altman, “Financial ratios, discriminant analysis and the prediction of corporate bankruptcy,” *The Journal of Finance*, vol. 23, no. 4, pp.

589–609, 1968. [Online]. Available: http://dx.doi.org/10.1111/j.1540-6261.1968.tb00843.x

[28] T. W. Anderson, “R. a. fisher and multivariate analysis,” *Statist. Sci.*, vol. 11, no. 1, pp. 20–34, 01 1996. [Online]. Available: http://dx.doi.org/10.1214/ss/1032209662

[29] ——, “Classification by multivariate analysis,” *Psychometrika*, vol. 16, no. 1, pp. 31–50, 1951. [Online]. Available: http://dx.doi.org/10.1007/BF02313425

[30] G. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley, 2004.

[31] M. R. Yousefi, J. Hua, C. Sima, and E. R. Dougherty, “Reporting bias when using real data sets to analyze classification performance,” *Bioinformatics*, vol. 26, pp. 68–76, 2010.

[32] U. Braga-Neto, A. Zollanvari, and E. Dougherty, “Cross-validation under separate sampling: strong bias and how to correct it,” *Bioinformatics*, vol. 30, no. 23, pp. 3349–3355, 2014.

[33] J. Hua, T. Waibhav, and E. R. Dougherty, “Performance of feature selection methods in the classification of high-dimensional data,” *Pattern Recogn.*, vol. 42, pp. 409–424, 2009.

[34] X. Chen, J. Higgins, S. T. Cheung, R. Li, V. Mason, and *et al.*, “Novel endothelial cell markers in hepatocellular carcinoma,” *Modern Pathol.*, vol. 17, pp. 1198–1210, 2004.

[35] C. Desmedt, F. Piette, S. Loi, Y. Wang, F. Lallemand, and *et al.*, “Strong time dependence of the 76-gene prognostic signature for node-negative breast cancer patients in the transbig multicenter independent validation series,” *Clin. Cancer Res.*, vol. 13, pp. 3207–3214, 2007.

[36] G. Natsoulis, L. E. Ghaoui, G. R. Lanckriet, A. M. Tolley, F. Leroy, and *et al.*, “Classification of a large microarray data set: algorithm comparison and analysis of drug signatures,” *Genome Res.*, vol. 1, pp. 724–736, 2005.

[37] A. Rosenwald, G. Wright, M. C. Chan, J. M. Connors, E. Campo, and *et al.*, “The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma,” *New Eng. J. Med.*, vol. 346, pp. 1937–1947, 2002.

[38] P. J. Valk, R. G. Verhaak, M. A. Beijen, C. A. Erpelinck, S. Barjesteh, and *et al.*, “Prognostically useful gene-expression profiles in acute myeloid leukemia,” *New Eng. J. Med.*, vol. 350, pp. 1617–1628, 2004.

[39] M. van de Vijver, Y. He, and et. al., “A gene-expression signature as a predictor of survival in breast cancer,” *N. Engl. J. Med.*, vol. 347, pp. 1999–2009, 2002.

[40] E. J. Yeoh, M. E. Ross, S. A. Shurtleff, W. K. Williams, D. Patel, and *et al.*, “Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling,” *Cancer Cell*, vol. 1, pp. 133–143, 2002.

[41] A. Azzalini and A. Capitanio, “Statistical applications of the multivariate skew-normal distribution,” *J. Roy. Statist. Soc. B*, vol. 61, pp. 579–602, 1999.

[42] F. Wyman, D. Young, and D. Turner, “A comparison of asymptotic error rate expansions for the sample linear discriminant function,” *Pattern Recogn.*, vol. 23, no. 7, pp. 775–783, 1990.

[43] D. Zhang, R. Zhu, H. Zhang, C. Zheng, and J. Xia, “Mgdb: a comprehensive database of genes involved in melanoma,” *Database: The Journal of Biological Databases and Curation*, vol. 2015:bav097, 2015.

[44] S. M. Agarwal, D. Raqhav, H. Singh, and G. P. Raghava, “Ccdb: a curated database of genes involved in cervix cancer.” *Nucleic Acids Res.*, vol. 39: D975-D979, 2010.

[45] L. C. Li, H. Zhao, H. Shiina, C. J. Kane, and R. Dahiya, “Pgdb: a curated and integrated database of genes related to the prostate,” *Nucleic Acids Res.*, vol. 1, no. 31, pp. 291–293, 2003.

[46] V. I. Serdobolskii, “Discriminant analysis of high-dimensional data over limited samples,” *Doklady Mathematics*, vol. 81, no. 1, pp. 75–77, 2010. [Online]. Available: http://dx.doi.org/10.1134/S1064562410010217

[47] S. Raudys and A. Saudargiene, “First-order tree-type dependence between variables and classification performance,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 233–239, Feb 2001.

[48] D. Bakir, A. P. James, and A. Zollanvari, “An efficient method to estimate the optimum regularization parameter in rlda,” *Bioinformatics*, 2016.

[49] A. R. Webb, *Statistical pattern recognition*. Wiley, 2002.

[50] V. N. Vapnik, *Statistical learning theory*, 1st ed. Wiley, Sep. 1998.

[51] A. I. Belousov, S. A. Verzakov, and J. von Frese, “A flexible classification approach with optimal generalization performance: Support vector machines,” *Chemometrics Intell. Lab. Syst.*, vol. 64, no. 1, pp. 15–25, 2002.

[52] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. J. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerson, “Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses,” *Proceedings of the National Academy of Sciences*, vol. 98, no. 24, pp. 13 790–13 795, 2001. [Online]. Available: http://www.pnas.org/content/98/24/13790.abstract

[53] B. K. Natarajan, “Sparse approximate solutions to linear systems,” *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995. [Online]. Available: http://dx.doi.org/10.1137/S0097539792240406

[54] J. Romberg, “Imaging via compressive sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, March 2008.

[55] D. Mackenzie and A. M. Society, “Compressed Sensing Makes Every Pixel Count,” in *What’s happening in the mathematical sciences*. AMS Bookstore, May 2009.

[56] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM Journal On Scientific Computing*, vol. 20, pp. 33–61, 1998.

[57] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. [Online]. Available: http://dx.doi.org/10.1111/j.1467-9868.2005.00503.x

[58] J. Friedman, T. Hastie, and R. Tibshirani, “Regularization paths for generalized linear models via coordinate descent,” *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v033i01

[59] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, Feb 2009.

[60] A. I. Su, J. B. Welsh, L. M. Sapinoso, S. G. Kern, P. Dimitrov, H. Lapp, P. G. Schultz, S. M. Powell, C. A. Moskaluk, H. F. Frierson, and G. M. Hampton, “Molecular classification of human carcinomas by use of gene expression signatures,” *Cancer Research*, vol. 61, no. 20, pp. 7388–7393, 2001. [Online]. Available: http://cancerres.aacrjournals.org/content/61/20/7388

[61] M. R. Yousefi and E. R. Dougherty, “Performance reproducibility index for classification,” *Bioinformatics*, vol. 28, pp. 2824–2833, 2012.

[62] M. Buyse, S. Loi, and et al., “Validation and clinical utility of a 70-gene prognostic signature for women with node-negative breast cancer,” *J. Natl. Cancer Inst.*, vol. 98, no. 17, pp. 1183–1192, 2006.

Appendicies

Appendix A

Bhattacharjee et al. [52] dataset describes different types of lung tumors collected from mRNA expression levels. In total, dataset contains 12600 gene-expression levels for 139 adenocarcinomas, 21 squamous cell lung carcinomas, 20 pulmonary carcinoids, 6 small-lung carcinomas and 17 normal lung (203 samples in total). The dataset was obtained from snap-frozen specimens and gene-expression levels were hybridized to human U95A oligonucleotide probe arrays (Affymetrix, Santa Carla, CA). For binary classification purpose, the tumors and normal lung labels were divided into two classes: 139 samples of adenocarcinomas are labeled as a single class, and all other labels (21 squamous cell lung carcinomas, 20 pulmonary carcinoids, 6 small-lung carcinomas, and 17 normal lung) are grouped to second class (64 samples in total).

Chen [34] database consists of 82 tumor and 75 non-tumor liver classes collected from Queen Mary Hospital, Stanford University and University of Hong Kong during surgical resections or transplants. The dataset were pre-proccesed to remove genes with more than 25% data missing resulting in 10237 genes left from original 24168.

Desmedt [35] dataset contains the gene profiles in frozen samples of 198 systematically untreated patients. The original data labeling included survival rates for less than 5 years and more than 5 years, and less than 10 years and more than 10 years which is not balanced for binary classification problem. The dataset labeling was changed to a patient survival rate for less than 10 years and more than 10 years, resuling in 77 patients for former class and 98 patients for the latter class.

Natsoulis [36] dataset describe the drug and toxicants test results conducted on rats. The male rats were continuously fed 22 different drugs and toxicants resulting in up to 12 tissues. To apply the database to a binary classification problem, 4 types of treatment were divided into two classes (strategy was adopted from [61]), the toxicant class of 61 samples and non-toxicant (fibrates 36, statin 31 and azoles 53 samples each) of 120 samples in total. The database is available at NIH Gene Expression Omnibus (GEO), the accession number is

GSE2187.

Rosenwald [37] microarray data consists of 240 samples of diffuse large-B-cell lymphoma and 12196 complementary DNA clones. The DNA clones constructed lymphochip DNA and were used to quantify mRNA expressions in the genes. The original database were clustered into three classes: germinal center B-cell like (115 samples), type 3 (52 samples) and activated B-cell like (73 samples) lymphomas. To apply the database in binary classification problems, [31] divided all the results into two sections according to survival rate within three years (alive or not) leading to 89 and 114 samples respectively. The features with more than 10% missing data were removed from database, and all other missing points were filled with corresponding feature mean values.

Su et al. [60] studied human carcinomas classification via human gene-expression profiles. The gene-expression profiles were assessed by H&E frozen section examination, the rich tumor area were cut from the frozen blocks. The complete processing of RNA extraction and hybridization (U95a GeneChip, Affymetix Incorporated, Santa Carla, CA) is described in [60]. The dataset consists of 174 sample points, 12533 features and 11 different tumor cells, for binary classification problem they were divided as follows: 8 bladder/ureter carcinomas, 26 infiltrating ductal breast adenocarcinomas, 23 colorectal adenocarcinomas, 26 prostate adenocarcinomas as class one (83 samples points in total), 12 gastroesophageal adenocarcinomas, 27 serous papillary ovarian adenocarcinomas, 11 clear cell carcinomas of kidney, 7 hepatocellular carcinomas, 6 pancreatic adenocarcinomas, 14 lung adenomacarcinomas carcinomas, and 14 lung squamous carcinomas as class two (91 samples points in total).

Valk [38] dataset contains 22215 features of 116 normal and 157 abnormal karyotype samples. The missing values were filled with the average values calculated feature wise across all samples. The dataset is publicly available at the NIG GEO under accession number GSE1159.

van der Vijver [39] dataset is constructed from breast cancer prognosis studies [39, 62] of gene-expression profile. There are 5003 human genes taken from breast cancer prognosis dataset of 180 and 115 samples from poor- and good-prognosis groups, where a poor-prognosis label predicts a distant metastasis within $5 - 10$ years of initial diagnosis.

Yeoh [40] database describes the pediatric acute lymphoblastic leukemia (ALL) disease, which has several subtypes. Original data was collected using Affymetric Human Genome HG U95Av2 array (Santa Clara, CA) and contained six labels: T-ALL (43 samples points), E2A-PBXI (27 sample points), TEL-AML1 (79 sample points), BCR-ABL (15 sample points), MLL (20 sample points), and hyperploid with more than 50 chromosomes (64 sample points). The data is freely distributed at: http://www.stjuderesearch.org/data/ALL1. Features with more 10% missing points were removed from database reducing the dataset dimension to 5077, the left missing points were filled with the mean value across all samples. For binary classification purposes, all labels were divided into two classes, BCR-ABL, MLL and hyperploid with more than 50 chromosomes belonging to one class (99 samples) and all other ALL subtypes (T-ALL, E2A-PBX1 and TEL-AML1) constructing the second class (149 samples).

Appendix B

This Appendix section shows the rank tables computed in Chapter 3. Each of the table show which classifier had the best performance compared to others.

Table B.1: Classifier ranking for** $p = 5$ **and** $p = 20$ **on Bhattacharjee Database

Classifiers	$p = 5$								$p = 20$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	6	6	6	6	6	6	6	6	9	9	7	7	7	3	2	2
DLDA	8	8	8	8	8	8	8	8	7	7	8	8	9	9	9	9
EDC	3	3	3	4	4	4	4	3	2	2	2	2	2	2	3	3
G13	10	10	10	10	10	10	10	10	10	10	10	10	8	8	8	8
RLDA	9	9	9	9	9	9	9	9	8	8	9	9	10	10	10	10
Serd	1	1	1	1	1	1	1	1	4	4	4	5	5	7	7	7
Serdc	2	2	2	2	2	2	2	2	3	3	3	4	4	6	6	6
LSVM	5	5	5	5	5	5	5	5	5	5	6	6	6	4	4	4
KSVM	4	4	4	3	3	3	3	4	1	1	1	1	1	1	1	1
Zar	7	7	7	7	7	7	7	7	6	6	5	3	3	5	5	5

Table B.2: Classifier ranking for* $p = 50$ *and* $p = 100$ *on Bhattacharjee Database

Classifiers	$p = 50$								$p = 100$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	9	9	9	9	9	6	6	6	10	10	9	9	10	10	10	10
DLDA	8	8	8	8	8	9	9	7	8	8	8	8	8	8	8	8
EDC	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
G13	10	10	10	10	10	10	10	10	9	9	10	10	9	9	9	9
RLDA	5	5	5	5	5	5	5	5	3	3	3	3	3	3	3	2
Serd	7	7	7	7	7	8	8	9	7	7	7	7	7	7	7	7
Serdc	6	6	6	6	6	7	7	8	6	6	6	6	6	6	6	6
LSVM	2	2	2	2	2	2	2	3	2	2	2	2	2	2	2	3
KSVM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Zar	4	3	3	3	3	3	3	2	5	4	4	4	4	4	4	4

Table B.3: Classifier ranking for* $p = 200$ *on Bhattacharjee Database

Classifiers	$p = 200$							
	30	40	50	60	70	80	90	100
LDA	9	9	10	10	10	10	9	9
DLDA	8	8	8	8	8	8	8	8
EDC	5	5	5	5	5	5	5	5
G13	10	10	9	9	9	9	10	10
RLDA	3	3	3	3	3	3	3	2
Serd	7	7	7	7	7	7	7	7
Serdc	6	6	6	6	6	6	6	6
LSVM	1	1	2	2	2	2	2	3
KSVM	2	2	1	1	1	1	1	1
Zar	4	4	4	4	4	4	4	4

Table B.4: Classifier ranking for $p = 5$ and $p = 20$ on Chen Database

Classifiers	$p = 5$								$p = 20$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	5	5	5	5	5	5	5	5	8	8	8	8	6	5	5	5
DLDA	4	4	4	4	4	4	4	4	5	6	6	6	8	8	8	8
EDC	1	1	1	1	2	2	2	2	3	5	5	5	7	7	7	7
G13	7	7	6	7	6	6	7	7	7	7	7	7	5	4	4	4
RLDA	2	3	3	3	3	3	3	3	6	4	4	4	4	6	6	6
Serd	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10
Serdc	10	10	10	10	10	10	10	10	9	9	9	9	9	9	9	9
LSVM	6	6	7	6	7	7	6	6	2	2	2	2	2	2	2	2
KSVM	3	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
Zar	8	8	8	8	8	8	8	8	4	3	3	3	3	3	3	3

Table B.5: Classifier ranking for $p = 50$ and $p = 100$ on Chen Database

Classifiers	$p = 50$								$p = 100$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	10	10	10	9	8	8	6	6	8	7	7	8	8	8	7	7
DLDA	6	6	6	6	6	6	8	8	6	6	6	6	6	6	6	6
EDC	5	5	5	5	5	5	7	7	5	5	5	5	5	5	5	5
G13	9	9	9	10	7	7	5	4	7	8	8	7	7	7	8	8
RLDA	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	3
Serd	7	7	8	7	9	9	10	9	10	10	10	10	10	9	9	9
Serdc	8	8	7	8	10	10	9	10	9	9	9	9	9	10	10	10
LSVM	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1
KSVM	2	2	2	2	2	2	2	1	3	3	3	3	3	3	3	2
Zar	4	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4

Table B.6: Classifier ranking for $p = 200$ on Chen Database

Classifiers	$p = 200$							
	30	40	50	60	70	80	90	100
LDA	9	10	10	10	10	9	9	9
DLDA	6	6	6	6	6	6	6	6
EDC	5	5	5	5	5	5	5	5
G13	10	9	9	9	9	10	10	10
RLDA	2	2	2	2	2	2	1	1
Serd	7	7	7	7	7	7	7	7
Serdc	8	8	8	8	8	8	8	8
LSVM	1	1	1	1	1	1	2	2
KSVM	3	3	3	3	3	3	3	3
Zar	4	4	4	4	4	4	4	4

Table B.7: Classifier ranking for $p = 5$ and $p = 20$ on Desmedt Database

Classifiers	$p = 5$								$p = 20$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	7	7	6	6	6	6	6	6	7	7	7	6	6	6	5	4
DLDA	3	4	4	7	7	7	7	7	2	4	5	5	5	5	6	7
EDC	1	1	1	1	1	2	3	3	1	1	4	4	4	4	4	5
G13	10	10	10	10	10	10	10	10	8	8	8	8	7	7	7	6
RLDA	9	8	8	8	8	8	8	8	4	3	2	3	3	3	3	3
Serd	2	2	2	2	4	5	5	5	10	10	10	10	10	10	10	10
Serdc	5	5	9	9	9	9	9	9	9	9	9	9	9	9	9	9
LSVM	6	6	5	4	3	1	1	1	6	6	6	7	8	8	8	8
KSVM	8	9	7	5	5	3	2	2	3	2	1	1	1	1	1	2
Zar	4	3	3	3	2	4	4	4	5	5	3	2	2	2	2	1

Table B.8: Classifier ranking for $p = 50$ *and* $p = 100$ *on Desmedt Database*

Classifiers	$p = 50$								$p = 100$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	10	10	9	7	7	7	7	7	9	10	10	9	9	9	10	10
DLDA	2	6	6	6	6	6	6	6	6	6	5	5	5	5	5	5
EDC	1	3	5	5	5	5	5	5	3	5	6	6	6	6	6	6
G13	9	9	10	8	8	8	8	8	10	9	9	10	10	10	9	9
RLDA	6	4	3	3	2	1	1	1	2	3	3	4	2	1	1	1
Serd	7	7	7	9	9	9	9	9	7	7	7	7	7	7	8	8
Serdc	8	8	8	10	10	10	10	10	8	8	8	8	8	8	7	7
LSVM	4	5	4	4	4	4	4	4	1	1	2	3	4	3	3	3
KSVM	3	1	1	2	3	3	3	3	4	2	1	1	1	2	2	2
Zar	5	2	2	1	1	2	2	2	5	4	4	2	3	4	4	4

Table B.9: Classifier ranking for $p = 200$ *on Desmedt Database*

Classifiers	$p = 200$							
	30	40	50	60	70	80	90	100
LDA	10	10	10	9	9	10	9	9
DLDA	6	5	5	5	5	5	5	5
EDC	5	6	6	6	6	6	6	6
G13	9	9	9	10	10	9	10	10
RLDA	2	2	2	2	2	1	1	1
Serd	8	8	7	8	8	7	8	8
Serdc	7	7	8	7	7	8	7	7
LSVM	1	1	1	1	1	2	2	2
KSVM	4	3	3	3	3	3	3	3
Zar	3	4	4	4	4	4	4	4

Table B.10: Classifier ranking for $p = 5$ ***and*** $p = 20$ ***on Natsoulis Database***

Classifiers	$p = 5$								$p = 20$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	7	7	7	7	7	7	7	6	9	9	7	7	5	5	5	4
DLDA	1	1	1	1	1	1	1	1	2	2	2	2	4	4	4	5
EDC	2	2	2	2	2	2	2	2	3	5	5	5	7	7	7	7
G13	10	10	9	9	8	8	8	8	10	10	10	10	8	8	8	8
RLDA	3	3	3	3	3	4	5	5	5	4	4	4	3	3	3	3
Serd	8	8	8	8	9	9	9	9	6	7	8	8	9	9	9	9
Serdc	9	9	10	10	10	10	10	10	7	8	9	9	10	10	10	10
LSVM	5	5	5	6	6	6	6	7	4	3	3	3	2	2	2	2
KSVM	4	4	4	4	4	3	3	3	1	1	1	1	1	1	1	1
Zar	6	6	6	5	5	5	4	4	8	6	6	6	6	6	6	6

Table B.11: Classifier ranking for $p = 50$ ***and*** $p = 100$ ***on Natsoulis Database***

Classifiers	$p = 50$								$p = 100$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	9	9	9	9	7	7	7	7	9	9	9	9	9	9	10	10
DLDA	3	3	4	4	4	4	4	4	3	3	4	4	5	5	5	5
EDC	4	5	5	5	6	6	6	6	4	5	5	6	6	6	6	6
G13	10	10	10	10	8	8	8	8	10	10	10	10	10	10	9	9
RLDA	5	4	3	3	3	3	3	3	5	4	3	3	3	3	3	3
Serd	7	7	7	7	9	9	9	9	7	7	7	7	7	7	7	7
Serdc	8	8	8	8	10	10	10	10	8	8	8	8	8	8	8	8
LSVM	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
KSVM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Zar	6	6	6	6	5	5	5	5	6	6	6	5	4	4	4	4

Table B.12: Classifier ranking for $p = 200$ ***on Natsoulis Database***

Classifiers	$p = 200$							
	30	40	50	60	70	80	90	100
LDA	10	10	10	10	9	9	10	9
DLDA	3	4	4	4	4	4	5	5
EDC	5	5	5	5	6	6	6	6
G13	9	9	9	9	10	10	9	10
RLDA	4	3	3	3	3	3	3	3
Serd	7	7	7	7	7	7	7	7
Serdc	8	8	8	8	8	8	8	8
LSVM	1	1	1	1	1	1	1	1
KSVM	2	2	2	2	2	2	2	2
Zar	6	6	6	6	5	5	4	4

Table B.13: Classifier ranking for $p = 5$ ***and*** $p = 20$ ***on Rosenwald Database***

Classifiers	$p = 5$								$p = 20$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	8	8	8	8	7	7	7	7	9	9	9	9	9	9	9	9
DLDA	2	2	2	2	2	2	2	2	3	3	3	3	3	3	4	4
EDC	1	1	1	1	1	1	1	1	4	4	4	4	4	5	5	5
G13	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
RLDA	6	5	5	4	3	3	3	3	6	6	6	6	6	6	6	6
Serd	3	3	3	3	4	4	4	4	1	1	1	1	1	1	1	1
Serdc	4	4	4	5	5	5	5	5	2	2	2	2	2	2	2	2
LSVM	9	9	9	9	9	9	9	9	8	8	8	8	8	8	8	8
KSVM	5	7	7	7	8	8	8	8	5	5	5	5	5	4	3	3
Zar	7	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7

Table B.14: Classifier ranking for $p = 50$ *and* $p = 100$ *on Rosenwald Database*

Classifiers	$p = 50$								$p = 100$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	10	10	9	9	9	9	9	9	9	10	10	10	9	9	9	9
DLDA	1	1	1	2	3	3	3	3	1	2	2	2	2	2	3	3
EDC	2	2	3	4	4	4	4	4	2	3	3	3	3	4	4	4
G13	9	9	10	10	10	10	10	10	10	9	9	9	10	10	10	10
RLDA	6	4	4	3	2	2	2	2	4	4	4	4	4	3	2	2
Serd	4	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6
Serdc	5	6	6	6	6	6	6	6	6	6	6	5	5	5	5	5
LSVM	7	8	8	8	8	8	8	8	7	7	7	7	7	8	8	8
KSVM	3	3	2	1	1	1	1	1	3	1	1	1	1	1	1	1
Zar	8	7	7	7	7	7	7	7	8	8	8	8	8	7	7	7

Table B.15: Classifier ranking for $p = 200$ *on Rosenwald Database*

Classifiers	$p = 200$							
	30	40	50	60	70	80	90	100
LDA	9	10	10	9	10	10	10	10
DLDA	1	2	2	2	2	2	2	3
EDC	3	3	3	3	3	3	3	4
G13	10	9	9	10	9	9	9	9
RLDA	4	4	4	4	4	4	4	2
Serd	6	7	7	7	6	6	7	7
Serdc	7	6	6	6	7	7	6	6
LSVM	5	5	5	5	5	5	5	5
KSVM	2	1	1	1	1	1	1	1
Zar	8	8	8	8	8	8	8	8

Table B.16: Classifier ranking for $p = 5$ ***and*** $p = 20$ ***on Valk Database***

Classifiers	$p = 5$								$p = 20$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	9	9	8	8	8	8	8	8	9	9	9	9	9	9	9	9
DLDA	5	5	5	6	6	6	7	7	3	3	3	3	3	3	4	4
EDC	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
G13	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
RLDA	6	6	6	5	5	4	4	4	4	4	4	4	4	4	3	3
Serd	2	2	3	3	3	3	3	3	5	5	5	5	5	6	6	6
Serdc	3	4	4	4	4	5	5	6	6	6	6	6	6	7	7	7
LSVM	7	7	7	7	7	7	6	5	8	8	8	8	8	8	8	8
KSVM	4	3	2	2	2	2	2	2	1	1	1	1	1	1	1	1
Zar	8	8	9	9	9	9	9	9	7	7	7	7	7	5	5	5

Table B.17: Classifier ranking for $p = 50$ ***and*** $p = 100$ ***on Valk Database***

Classifiers	$p = 50$								$p = 100$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	10	10	9	9	9	9	9	9	9	10	10	9	9	9	9	9
DLDA	3	3	3	4	4	4	4	5	3	3	4	4	4	4	5	5
EDC	2	2	2	3	3	3	3	3	2	2	2	2	3	3	3	4
G13	9	9	10	10	10	10	10	10	10	9	9	10	10	10	10	10
RLDA	4	4	4	2	2	2	2	2	4	4	3	3	2	2	2	2
Serd	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
Serdc	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
LSVM	5	6	6	6	6	6	6	6	5	5	6	6	6	6	6	6
KSVM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Zar	6	5	5	5	5	5	5	4	6	6	5	5	5	5	4	3

Table B.18: Classifier ranking for $p = 200$ ***on Valk Database***

Classifiers	$p = 200$							
	30	40	50	60	70	80	90	100
LDA	10	9	9	9	10	10	9	9
DLDA	3	3	4	4	4	4	4	5
EDC	2	2	2	3	3	3	3	3
G13	9	10	10	10	9	9	10	10
RLDA	4	4	3	2	2	2	2	2
Serd	7	7	7	7	7	7	7	7
Serdc	8	8	8	8	8	8	8	8
LSVM	5	5	5	5	6	6	6	6
KSVM	1	1	1	1	1	1	1	1
Zar	6	6	6	6	5	5	5	4

Table B.19: Classifier ranking for $p = 5$ ***and*** $p = 20$ ***on Vijver Database***

Classifiers	$p = 5$								$p = 20$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	4	4	4	4	4	3	3	3	8	8	8	7	6	6	6	6
DLDA	7	7	7	7	7	7	7	7	3	3	4	4	4	4	4	4
EDC	3	3	3	3	3	4	5	5	2	2	2	2	2	2	2	3
G13	9	9	9	9	10	10	10	10	10	10	10	10	10	10	10	10
RLDA	10	10	10	10	9	9	9	9	9	9	9	9	9	9	9	9
Serd	6	6	6	6	6	6	6	6	5	6	6	6	7	7	7	7
Serdc	8	8	8	8	8	8	8	8	7	7	7	8	8	8	8	8
LSVM	2	2	2	2	2	2	1	1	4	4	3	3	3	3	3	2
KSVM	1	1	1	1	1	1	2	2	1	1	1	1	1	1	1	1
Zar	5	5	5	5	5	5	4	4	6	5	5	5	5	5	5	5

Table B.20: Classifier ranking for $p = 50$ and $p = 100$ on Vijver Database

Classifiers	$p = 50$								$p = 100$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	9	10	9	9	8	8	8	6	10	9	9	9	9	9	9	10
DLDA	4	4	3	3	3	3	2	2	4	4	3	2	2	2	2	2
EDC	3	2	2	2	2	2	3	4	3	3	2	3	3	3	3	3
G13	10	9	10	10	10	10	10	10	9	10	10	10	10	10	10	9
RLDA	8	8	8	8	9	9	9	9	8	8	8	8	8	8	8	8
Serd	6	6	6	6	6	6	6	7	6	6	6	6	6	6	6	6
Serdc	7	7	7	7	7	7	7	8	7	7	7	7	7	7	7	7
LSVM	2	3	4	5	5	5	5	5	2	2	4	4	4	5	5	5
KSVM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Zar	5	5	5	4	4	4	4	3	5	5	5	5	5	4	4	4

Table B.21: Classifier ranking for $p = 200$ on Vijver Database

Classifiers	$p = 200$							
	30	40	50	60	70	80	90	100
LDA	10	9	10	10	9	9	9	10
DLDA	4	4	4	4	4	4	4	4
EDC	3	3	3	3	3	3	3	3
G13	9	10	9	9	10	10	10	9
RLDA	8	8	8	8	8	8	8	8
Serd	6	6	6	6	6	6	6	6
Serdc	7	7	7	7	7	7	7	7
LSVM	1	2	2	2	2	2	2	2
KSVM	2	1	1	1	1	1	1	1
Zar	5	5	5	5	5	5	5	5

Table B.22: Classifier ranking for $p = 5$ *and* $p = 20$ *on Yeoh Database*

Classifiers	$p = 5$								$p = 20$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	9	7	7	5	5	5	5	5	7	7	7	5	5	4	3	3
DLDA	5	5	5	6	6	6	7	7	6	6	6	7	8	8	8	8
EDC	3	3	3	3	3	3	4	4	2	4	4	4	4	5	5	5
G13	10	10	10	10	10	10	10	10	8	8	8	8	6	6	6	6
RLDA	4	6	6	7	7	8	9	9	5	5	5	6	7	7	7	7
Serd	2	2	2	2	2	2	2	2	9	9	9	9	9	9	9	9
Serdc	1	1	1	1	1	1	1	1	10	10	10	10	10	10	10	10
LSVM	7	8	9	9	8	7	6	6	3	3	3	3	3	3	4	4
KSVM	6	4	4	4	4	4	3	3	1	1	1	1	1	1	1	1
Zar	8	9	8	8	9	9	8	8	4	2	2	2	2	2	2	2

Table B.23: Classifier ranking for $p = 50$ *and* $p = 100$ *on Yeoh Database*

Classifiers	$p = 50$								$p = 100$							
	30	40	50	60	70	80	90	100	30	40	50	60	70	80	90	100
LDA	10	9	9	10	7	7	7	7	10	10	10	9	9	9	10	10
DLDA	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
EDC	3	4	5	5	5	5	5	5	3	5	5	5	5	5	5	5
G13	9	10	10	9	8	8	8	8	9	9	9	10	10	10	9	9
RLDA	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Serd	7	7	7	7	9	9	9	9	7	7	7	7	7	7	7	7
Serdc	8	8	8	8	10	10	10	10	8	8	8	8	8	8	8	8
LSVM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2
KSVM	4	3	3	2	2	2	2	2	5	3	3	3	3	3	3	3
Zar	2	2	2	3	3	3	3	3	2	2	2	2	2	2	1	1

Table B.24: Classifier ranking for $p = 200$ ***on Yeoh Database***

Classifiers	$p = 200$							
	30	40	50	60	70	80	90	100
LDA	10	10	10	10	10	10	10	9
DLDA	6	6	6	6	6	6	6	6
EDC	4	4	4	5	5	5	5	5
G13	9	9	9	9	9	9	9	10
RLDA	3	3	3	3	3	3	3	3
Serd	7	7	7	7	7	7	7	7
Serdc	8	8	8	8	8	8	8	8
LSVM	1	1	1	1	1	1	1	1
KSVM	5	5	5	4	4	4	4	4
Zar	2	2	2	2	2	2	2	2

Appendix C

This section describes how to calculate the inverse covariance matrix of Zarutskij Classifier $\mathbf{\Sigma}^{-1}_{\text{Tree}}$. Let $\mathbf{\Sigma}$ be a covariance matrix given in Table C.1

Table C.1: Covariance Matrix

0.746	0.175	0.498	0.275	-0.153
0.175	0.966	0.332	0.344	-0.229
0.498	0.332	0.646	0.232	-0.299
0.275	0.344	0.232	0.693	-0.234
-0.153	-0.229	-0.299	-0.234	0.577

The correlation matrix is calculated by $r_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$, where $\mathbf{\Sigma} = \{\sigma_{i,j}\}$. The computed correlation matrix is given in Table C.2: From Table C.2 we see that

Table C.2: Correlation Matrix

1.000	0.207	0.717	0.383	-0.233
0.207	1.000	0.420	0.421	-0.307
0.717	0.420	1.000	0.347	-0.490
0.383	0.421	0.347	1.000	-0.371
-0.233	-0.307	-0.490	-0.371	1.000

$\mathbf{m} = \{1, 1, 1, 2, 3\}$, meaning that there will be three branches going into r_1 and one branch going into r_3 where r_i are the branches. The total amount of branches is the same as the number of features in the data, and correlation matrix values are the weights between the branches. Select $i = 2$, then $\{2, m_i\} = \{2, m_1\} = \{2, 1\}$. Calculate $\mathbf{C} = \{c_{i,j}\}$ according to this:

$$c_{ij} = \begin{cases} \left(\hat{\sigma}_{ii}\left(1 - r_{im_i}^2\right)\right)^{-\frac{1}{2}} & \text{if } j = i \\ \dfrac{-r_{im_i}}{\sqrt{\sigma_{m_im_i}\left(1 - r_{im_i}^2\right)}} & \text{if } j = m_i \\ 0 & \text{if otherwise,} \end{cases} \tag{5.1}$$

Table C.3: Lower Triangle Matrix

1.000	0.000	0.000	0.000	0.000
-0.245	1.040	0.000	0.000	0.000
-1.190	0.000	1.783	0.000	0.000
0.000	-0.472	0.000	1.324	0.000
0.000	0.000	0.699	0.000	1.511

C is given in Table C.3.
The inverse covariance matrix is computed by $\mathbf{\Sigma}_{\text{Tree}}^{-1} = \mathbf{C}^T\mathbf{C}$ and result is shown below:

Table C.4: Total Matrix

2.475	-0.254	-2.121	0.000	0.000
-0.254	1.305	0.000	-0.625	0.000
-2.121	0.000	3.668	0.000	1.056
0.000	-0.625	0.000	1.753	0.000
0.000	0.000	1.056	0.000	2.282

Appendix D

This section provides the pseudo-codes of simulations and algorithms used in Chapters 2 (all pseudo-codes) and Chapter 3 (γ^{opt} estimation procedure).

Algorithm 1: Estimation of Optimum Regularization Parameter $\left(\gamma^{\text{opt}}\right)$

```
1 select a maximum testing $\gamma$ as $\gamma_{\max}$;
2 select amount of $\gamma$ to test as $n_\gamma$;
3 compute $\gamma_{\text{base}} = (\gamma_{\max})^{\frac{1}{n_\gamma}}$;
4 let the array $\boldsymbol{\gamma}$ to test be $\left\{\gamma_{\text{base}}^{-n_\gamma+i}\right\}$ for $i = 0, 1, 2, \ldots, 2n_\gamma$;
5 for each $\gamma$ in $\boldsymbol{\gamma}$ do
6     estimate expected true error using (2.8)-(2.11) and (2.12);
7 end
8 estimate $\gamma_{\text{opt}}$ as the one that has the lowest expected true error;
```

Next pseudo-code describes the dataset sampling algorithm used for Chapter 2

Algorithm 2: Dataset Sampling Algorithm

```
1 load dataset;
2 if all features are 0 || any NaN is detected then
3     remove a detected feature vectors;
4 end
5 compute necessary variables $r$, $\alpha_0$ and $\alpha_1$;
6 compute $p$-values using two-sample t-test, and sort them in ascending
    order;
7 select the first $p$ indexes;
8 generate indexes for training and testing samples for each repetition;
9 save new low-dimensional database, and generated sample indexes;
```

Pseudo-codes used to estimate the expected true error of RLDA and its true error in Chapter 2 using real and synthetic databases (not coincidence is given in brackets).

Algorithm 3: Expected True Error and True Error Computation Algorithm Used in Chapter 2

```
1  load sampled dataset (do nothing);
2  assign necessary values $r$, $alpha_0$ $alpha_1$, etc;
3  for each repetition do
4      retrieve training and testing samples (generate training and testing
         samples);
5      calculate $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{C}$;
6      save time stamp;
7      apply each estimator and save time stamp after each of them;
8      compute true error of RLDA using hold-out estimator (compute true
         error of RLDA, for skewed-normal use hold-out estimator);
9      save time stamp;
10     calculate time differences;
11 end
12 save all the results;
13 generate figures;
```

Appendix E

Pseudo-code of simulations used in Chapter 3.

Algorithm 4: Computation of True Error in Chapter 3

```
1  load dataset;
2  assign necessary values $p$, $n$, $r$, etc;
3  if data is not lognormal then
4      transform dataset to a lognormal space;
5  end
6  normalize data feature size;
7  apply two-sample ttest to compute the best features indexes;
8  for each repetition do
9      generate training and testing samples;
10     compute the true error of each classifier;
11 end
12 save all the results;
13 generate figures;
```

Appendix F

Pseudo-code of simulations used in Chapter 4.

Algorithm 5: Computation of glment and SRC True Error in Chapter 4

```
1  load dataset;
2  assign necessary values $p$, $n$, $r$, etc;
3  for each repetition do
4      generate training and testing samples;
5      apply glmnet classifier;
6      generate randomgenes for each feature size of SRC;
7      for each feature size of SRC do
8          generate dimensionality reduced dataset;
9          apply CV10F-10R and compute the expected true error;
10     end
11     retrieve the feature size with the lowest expected true error;
12     compute the true error of SRC using a corresponding optimal
         dimension and randomgenes;
13 end
14 save all the results;
15 generate figures;
```