Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

D I P L O M A R B E I T

Classification in high-dimensional feature spaces

Ausgeführt am Institut für

Statistik und Wahrscheinlichkeitstheorie

der Technischen Universität Wien

unter der Anleitung von Professor Peter Filzmoser

durch

Fabian Schroeder

Rudolfsplatz 2/32, 1010 Wien

17.09.2012

Abstract

The characteristic property of many data sets in modern scientific fields, such as genomics, is the high-dimensionality of its feature space. It poses a significant challenge for statistical methods for classification and has thus been the object of intensive research in the past decade. This work studies the different approaches, with which standard classification methods, such as *Discriminant Analysis*, *Support Vector Machines* and *Logistic Regression*, have been modified to account for high-dimensionality, and compares their performance in different simulation experiments. Both the prediction as well as the model selection performance are examined under different parameters, including sample size, signal-to-noise ratios, and different structures of dependence. The results are supposed to guide the applied researcher in one of the most tricky questions: Choosing the most suitable method for a given research question and data set.

Zusammenfassung

Die charakteristische Eigenschaft vieler Daten aus modernen Wissenschaften wie z.B. der Genetik ist die Vielzahl an Variablen. Dies stellt eine große Herausforderung für statistische Verfahren der Klassifikation dar und wurde in den letzten Jahren intensiv untersucht. In dieser Arbeit soll studiert werden, wie die klassischen Methoden der Klassifikation, die Diskriminanzanalyse, die Support Vector Machine und die Logistische Regression, für die Anwendung auf hochdimensionale Räume modifiziert werden kann. In mehreren Simulationsexperimenten sollen sowohl die Güte der Prognose als auch der Modellselektion miteinander verglichen werden. Dabei wurden verschiedene Parameter wie Stichprobengröße oder das Verhältnis von Signal und Noise und verschiedenen Strukturen der Abhängigkeit variiert. Die Resultate sollen die Wahl der richtigen Methode für eine konkrete Fragestellung und einen konkreten Datensatz unterstützen.

Contents

1 Introduction **3**
1.1 The Model . . . 4
1.2 Classification . . . 6

2 Discriminant Analysis **7**
2.1 Quadratic Discriminant Analysis . . . 8
2.2 Linear Discriminant Analysis . . . 9
2.3 Regularized Discriminant Analysis . . . 10
2.4 Comparing Different Discrimination Rules . . . 10
2.5 Discriminant Analysis in High-Dimensional Feature Spaces . . . 11
2.6 Independence Rule . . . 12
2.7 Feature Annealed Independence Rule . . . 13
2.8 Nearest Shrunken Centroids . . . 14
2.9 Shrunken Centroids Regularized Linear Discriminant Analysis . . . 15

3 Support Vector Machines **17**
3.1 Geometrical View of SVMs . . . 17
3.2 SVMs as a Solution to a Tikhonov Regularized Optimization Problem . . . 19
3.3 Support Vector Machines in High-Dimensional Feature SSpaces . . . 19
3.4 Multi-Class Classification . . . 20

4 Logistic Regression **20**
4.1 Comparing Linear Discriminant Analysis and Logistic Regression . . . 21
4.2 Logistic Regression in High-Dimensional Feature Spaces . . . 22

5 Implementations **22**

6 Simulation Experiments **23**
6.1 Settings A - C: Independent Features . . . 24
6.2 Settings D - F: Block-Dependent Features . . . 24
6.3 Settings G - I : Randomly Dependent Features . . . 25
6.4 Results . . . 25

7 Real Data Analysis: The COPD Data **26**
7.1 The Two-Class Case . . . 33
7.2 The Three-Class Case . . . 34
7.3 The Five-Class Case . . . 35

8 Conclusion **36**

9 Appendix **37**
9.1 R Code: Simulation . . . 37
9.2 R Code: FAIR . . . 41
9.3 R Code: IR . . . 42
9.4 R Code: Wrapper Functions . . . 43
9.5 R Code: Cross Validation . . . 44

1 Introduction

The development of modern sequencing machines and DNA chip technologies have made it possible to measure the gene expression levels of all known gene products in a matter of days in a highly parallel manner. This technological breakthrough allows to conduct simple experiments that seek to identify those genes which are responsible for certain phenotype, e.g. a disease. The usual setup of such an experiment is as follows: The gene expression values of biological samples are measured in different conditions, e.g. transbronchial biopsies in healthy and diseased patients, in large numbers and are compared as to identify all those genes that play an important role in the studied phenomenon. Those genes that show a significant difference between conditions are said to be differentially expressed, which serves as an indicator for some function in the observed condition. Once the set of active genes is identified it is possible to further investigate these mechanism and to gain a deeper understanding about the phenomenon.

The role of statistics in this undertaking is to develop sound methods which can evaluate and describe the information of high-throughput technologies and possibly to build models that can predict certain conditions based solely on gene expression data. The recent technological advances, however, also come with (statistical) challenges. The most characteristic feature of gene expression data is its high dimensionality. Usually the number of features (genes) greatly outnumber the number of observations. In the conducted experiment the number of features is around 20000 after preprocessing, which already filters out many genes, whereas the number of observations is around 150. This does not only render many statistical models infeasible but it also induces spurious correlation with the response. Many features will be highly correlated with the response by pure chance.

The object of this study is a progressive disease of the lungs called *Chronic Obstructive Pulmonary Disease* (COPD). It is associated with heavy smoking or long-term exposure to other lung irritants such as air pollution, chemical fumes, or dust. COPD is a major cause of disability, and it is the third leading cause of death in the United States with rising numbers[1]. There currently exists no cure for COPD and the diagnosis relies on a complex set of clinical parameters. The degree of COPD has been categorized by pulmonologists with the Gold classification ranging from 0 (healthy) to 4 (severe COPD). This categorization is based on a large number of tests for diagnosis including exercise tests, lung function tests, blood examination, and computer tomography. The experiments were conducted under the supervision of Professor Ziesche at the medical university of Vienna between January 2009 and March 2012. The gene expression values were measured with Affymetrix chips type Human Genome U133 Plus 2.0[2].

The purpose of this work is to build a classification model for COPD GOLD grades based on gene expression data. The model should on the one hand have good predictive power, such that it might be used as a diagnosis tool for future patients, and should on the other hand allow a biological interpretation. The model building process is influenced by assumptions that are of genetical origin. Therefore section 1.1 introduces a statistical model for this kind of data, introducing basic notation and illustrating how these assumptions are modeled. Section 1.2 will inaugurate the general classification problem, providing a general framework for the subsequent methods. Several model families, that have been successfully applied to gene-expression data in the past, were applied to the COPD data set. These model families include *Discriminant Analysis*, *Support Vector Machines*, and *Logistic Regression*, introduced in sections 2, 3 and 4, respectively. First the standard models will be established, and then it will be shown

[1] http://www.nhlbi.nih.gov/health/health-topics/topics/copd/

[2] http://media.affymetrix.com/support/technical/datasheets/hgu133arrays_datasheet.pdf

how these models can be modified in order to account for high-dimensionality. Section 2, for instance, will illustrate how the ideas of the classical *Linear Discriminant Analysis* (LDA), the *Quadratic Discriminant Analysis* (QDA) and the *Regularized Discriminant Analysis* (RDA) can be adapted for high-dimensional feature spaces yielding the *Independence Rule* (IR), the *Nearest Shrunken Centroids* method (NSC) and the *Shrunken Centroids Regularized Discriminant Analysis* (SCRDA). The measures include strong assumptions concerning the dependence structure and feature selection. In section 3 the *Support Vector Machine* (SVM) is introduced. Its original form is a two-class classifier, however it can be extended to serve as a multi-class classifier. Regularization allows the application to high-dimensional feature spaces. This approach was also used to modify *Logistic Regression* (LR) in order to obtain sparse models, see section 4. The SVM and LR share a common framework, see section 3.2. Also LDA and LR can be compared since they both model posterior probabilities, see section 4.1.

Although all methods take a different approach in accounting for the dimensionality of the feature space, they can be interpreted as an attempt to reduce the prediction variance in return for accepting a small bias. Thus, this work concerns one the classic stories in statistics: the bias-variance tradeoff.

The simulation experiments in section 6 will analyze both the prediction and the model selection performance of the proposed methods in a myriad of different settings. From the results it should be possible to learn which methods work under which circumstances. All methods will then be applied to the COPD data set in section 7. This final conclusion can be found in section 8.

1.1 The Model

Gene expression values are modeled to consist of a systematic part μ and a random part ϵ

$$X_{ijk} = \mu_{jk} + \epsilon_{ijk} \qquad 1 \leq i \leq n_k, 1 \leq j \leq p, k = 1, \ldots, K, \tag{1}$$

where X_{ijk} denotes the expression value of the ith sample of the jth feature in class $\mathcal{C}_k$. The mean effect of the jth feature in class $\mathcal{C}_k$ is denoted by μ_{jk} and is supposed to capture the difference in expected expression values between the members of different groups. The error terms $\epsilon_{1j1}, \ldots, \epsilon_{n_1 j1}, \epsilon_{1j2}, \ldots, \epsilon_{n_2 j2}, \ldots, \epsilon_{n_k jk}$ are assumed to be i.i.d. with $E(\epsilon_{ijk}) = 0$. Throughout this work i will denote the index of observations, where i can be of any class $\mathcal{C}_k = \{1, \ldots, n_k\}$ for $k = 1, \ldots, K$. The overall number of observations is $n = \sum_{k=1}^{K} n_k$. The index of features will be denoted by j, ranging from $1, \ldots, p$, and k will denote the index of classes ranging from $1, \ldots, K$. The sample observations are summarized in the matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}'_{1\cdot 1} \\ \vdots \\ \boldsymbol{x}'_{n_1 \cdot 1} \\ \boldsymbol{x}'_{1\cdot 2} \\ \vdots \\ \vdots \\ \boldsymbol{x}'_{n_k \cdot k} \end{bmatrix} \in \boldsymbol{R}^{n \times p},$$

where e.g. $\boldsymbol{x}'_{1\cdot 1}$ is the transposed first observation of class 1. The observed class memberships are categorical and are summarized in the vector

$$\boldsymbol{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} \in \{1, \ldots, K\}^n.$$

Our current understanding of the genome allows to make certain assumptions. The first widely accepted assumption is that only a small fraction of measured genes play a functional role in any condition, e.g.the different classes of the disease COPD. In more technical terms: observations are identically distributed within each class with a distribution function F_k

$$X_{1 \cdot k}, \ldots, X_{n_k \cdot k} \sim F_k \tag{2}$$

and for most genes the average expression value, which is a component of the expectation of the random variables, does not differ between classes. Let s denote the number of genes that are differentially expressed between classes and without loss of generality assume that this is the case for the first s genes. This assumption can be modeled by

$$\mu_{j1} = \cdots = \mu_{jK} \qquad \forall j = s, \ldots, p \tag{3}$$

$$\exists (k, l) : \mu_{jk} \neq \mu_{jl} \qquad \forall j = 1, \ldots, s$$

where s will be very small compared to p. Thus, the classification model is supposed to be sparse in the sense that the prediction is only based on a handful of genes. This will also allow a more thorough biological interpretation. Many of the classification methods, that will be introduced, therefore involve feature selection.

Another important assumption, that is based on available knowledge about the genome, is that our genes are ordered in groups, called pathways, which serve a common function in our body. Therefore the expression values of genes in the same group should be highly correlated whereas gene expression values of genes in different groups should be independent. Although it is true that weak connections between groups may exist, independence between groups is usually a reasonable assumption. Within these pathways genes may influence its function by activation or inhibition such that the correlation can be positive or negative. The correlation levels also depend of the relative distance of two genes in the regulatory pathway. The further apart two genes are, the less correlation will be between them. These assumptions can be expressed in terms of the dependence structure of the genes. The covariance matrix of the features of our data set should also be sparse in the sense that most values are (close to) zero and that it has block structure, where any block is associated to a regulatory pathway, e.g.

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_1 & 0 & \cdots & 0 & 0 \\ 0 & \boldsymbol{\Sigma}_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\Sigma}_{m-1} & 0 \\ 0 & 0 & \cdots & 0 & \boldsymbol{\Sigma}_m \end{pmatrix}. \tag{4}$$

One example for a dependence structure of this kind is used in the simulation study. The correlation structure is assumed to be the same for every block with alternating signs. Further, the correlation structure in every block is assumed to be autoregressive, meaning that the correlation matrix has the following structure:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_\rho & 0 & \cdots & 0 & 0 \\ 0 & \boldsymbol{\Sigma}_{-\rho} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\Sigma}_\rho & 0 \\ 0 & 0 & \cdots & 0 & \boldsymbol{\Sigma}_{-\rho} \end{pmatrix}, \tag{5}$$

with blocks of the following structure:

$$\boldsymbol{\Sigma}_\rho = \begin{pmatrix} 1 & \rho & \cdots & \rho^{98} & \rho^{99} \\ \rho & 1 & \cdots & \rho^{97} & \rho^{98} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho^{98} & \rho^{97} & \cdots & 1 & \rho \\ \rho^{99} & \rho^{98} & \cdots & \rho & 1 \end{pmatrix}. \tag{6}$$

1.2 Classification

Consider the problem of assigning an object (e.g. a patient) with an unknown class membership to one of several classes $\mathcal{C}_1, \ldots, \mathcal{C}_K$ (e.g. COPD classes) based on a set of measurements $X = (X_1, \ldots, X_p)$ of that object (e.g. gene expression values). Every object is assumed to be member of one and one class only. A discrimination or classification rule $\delta(\boldsymbol{x})$ is supposed to be developed from a training set $(c_i, \boldsymbol{x}_i)_{i=1,\ldots,n}$ of observations $\boldsymbol{x}_i \in \mathbb{R}^p$ with a known class membership $c_i \in \{1, \ldots, K\}$ and then applied to a test set of observations with the objection of predicting the unknown class memberships. This approach is referred to as *supervised learning*.

A (K-class) *classifier* is a mapping

$$\delta : \Omega \to \{1, \ldots, K\},$$

where $\Omega \subseteq \mathbb{R}^p$ is the sample space. Most classifiers are of the form

$$\delta(\boldsymbol{x}) = \max_{k=1,\ldots,K} \delta_k(\boldsymbol{x}), \tag{7}$$

where δ_k is a mapping $\delta_k : \Omega \to \mathbb{R}$, sometimes referred to as the *class discriminant*. A *linear classifier* is defined by the characteristic that the classification is based on a linear combination of the characteristics,

$$\delta(\boldsymbol{x}) = \max_{k=1,\ldots,K} f(\boldsymbol{w}_k \cdot \boldsymbol{x}), \tag{8}$$

where $\boldsymbol{w} \in \mathbb{R}^p$ is a vector of coefficients defining the linear boundary between any two classes and f is a real-valued mapping. This approach allows a geometric interpretation. It divides the sample space $\boldsymbol{\Omega}$ into K disjoint regions $R_1, \ldots, R_K$. R_i can be defined as

$$R_i := \{\boldsymbol{x} \in \boldsymbol{\Omega} : \delta(\boldsymbol{x}) = i\}. \tag{9}$$

Using this notation the classifier (7) can be expressed as

$$\delta(\boldsymbol{x}) = k \text{ iff } \boldsymbol{x} \in R_k.$$

The boundaries between R_i and R_j are those points in $\boldsymbol{\Omega}$ that satisfy

$$\{\boldsymbol{x} \in \boldsymbol{\Omega} : \max_{k=1,\ldots,K} \delta_k(\boldsymbol{x}) = \delta_i(\boldsymbol{x}) = \delta_j(\boldsymbol{x})\},$$

which will be linear or non-linear, depending on the discrimination function. In order to obtain linear boundaries it suffices to demand that some monotone transformation of the decision rule is linear. Most discrimination methods that will be used in this work will be linear in this sense.

A classical approach in statistics to derive a classification rule is to minimize the *expected prediction error* or *expected loss*

$$EPE = E_{(\mathcal{C},X)}[L(\mathcal{C}, \delta(X))], \tag{10}$$

where $L(\mathcal{C}, \delta(X))$ is a *loss function* and the expectation is taken with respect to the joint distribution $P(\mathcal{C}, X)$. Since the number of classes is finite the loss function can be represented by a $K \times K$ matrix L

$$L(\mathcal{C}, \delta(X)) = (l_{ij})_{\substack{i=1,\ldots,K \\ j=1,\ldots,K}} \text{ if } \mathcal{C} = i, \delta(X) = j. \tag{11}$$

The diagonal of the matrix L consists of zeros since a correctly classified object should cause no loss. The remaining elements l_{ij} can be interpreted as the disutility of incorrectly classifying an object of class $\mathcal{C}_i$ as a member of class $\mathcal{C}_j$ and should therefore be non-negative. This general form of a loss function allows to weight misclassification asymmetrically e.g. the misclassification of ill patient as healthy more severely than the misclassification of a healthy patient as ill. The simplest loss function is characterized by an equal valuation of every misclassification, yielding

$$l_{ij} = \begin{Bmatrix} 0 & \text{if} & i = j \\ 1 & \text{if} & i \neq j \end{Bmatrix}. \tag{12}$$

This most natural loss function amounts to simply counting the number of misclassified cases and is called the *Misclassification Error*. The theoretical *Misclassification Error* (MCE) of a discrimination rule δ can be written as

$$MCE(\delta) = \sum_{k=1}^{K} P[\delta(X) \neq k, \mathcal{C} = k], \tag{13}$$

where P is the common distribution of (X, C). The *Empirical Classification Error* or *Misclassification Rate* based on a sample of n observations is simply the rate of incorrectly classified samples

$$mce(\delta) = \frac{1}{m} \sum_{i=1}^{n} \mathbb{1}\{\delta(\boldsymbol{x}_i) \neq c_i\}. \tag{14}$$

Throughout this work preferably the *rate of correctly classified observations*

$$rcc(\delta) = 1 - mce(\delta)$$

will be used. Needless to say, that this statistic can be used to compare the prediction power of different classifiers.

2 Discriminant Analysis

The family of methods called *Discriminant Analysis* (DA) can be derived by conditioning (10) on X yielding

$$EPE_X = E_X[\sum_{k=1}^{K} L(\mathcal{C}, \delta(X))P(\mathcal{C} = k|X)], \tag{15}$$

and modeling the conditional density $P[\mathcal{C}|X]$. This decision theory of methods of DA is mainly based on chapters 2.4 and 4 of Hastie et al. (2009). Minimizing expression (15) will yield a discrimination rule for an observation $\boldsymbol{x}$

$$\delta(\boldsymbol{x}) = \arg\min_{c=1,\ldots,K} \sum_{k=1}^{K} L(\mathcal{C} = k, c)P(\mathcal{C} = k|X = \boldsymbol{x}).$$

Using the loss function (12) yields

$$\delta(\boldsymbol{x}) = \arg\min_{k=1,\ldots,K}[1 - P(\mathcal{C} = k|X = \boldsymbol{x})] = \arg\max_{k=1,\ldots,K} P(\mathcal{C} = k|X = \boldsymbol{x}) \tag{16}$$

called the Bayes classifier. The associated error rate is called the Bayes rate.

From expression (16) we can see that for optimal classification in the Bayesian sense it is necessary to know the class posteriors $P(\mathcal{C}|X)$. Therefore let us assume that the observations of group $\mathcal{C}_k$ are independent and follow a common distribution $f_k(\boldsymbol{x})$, called class-conditional distribution. Let π_k denote the prior probability of class k, with $\sum_{k=1,\ldots,K} \pi_k = 1$. Then Bayes theorem tells us that the class-posterior distribution is

$$P(\mathcal{C} = k|X = \boldsymbol{x}) = \frac{f_k(\boldsymbol{x})\pi_k}{\sum_{l=1,\ldots,K} f_l(\boldsymbol{x})\pi_l}.$$

If the class conditional distribution is known, then it is possible to calculate the expected loss (10) and to derive a classification function minimizing the expected loss.

$$EPE(\delta, \boldsymbol{x}) = \frac{\sum_{k=1,\ldots,K} L(\mathcal{C}_k, \delta(\boldsymbol{x}))f_k(\boldsymbol{x})\pi_k}{\sum_{l=1,\ldots,K} f_l(\boldsymbol{x})\pi_l}$$

Therefore, assumptions regarding the class-conditional distribution will characterize the discrimination methods. The assumption of Gaussian densities $f_k \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ will lead to a method called Quadratic Discriminant Analysis (QDA) explained in section 2.1. Assuming that all classes share a common covariance structure $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, k = 1, \ldots, K$, will yield a method called Linear Discriminant Analysis (LDA), see section 2.2. Assuming that the features are independent, which (for Gaussian class-conditional distributions) amounts to the assumption that $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_p)$ will result in a method called Naive Bayes, see section 2.6.

2.1 Quadratic Discriminant Analysis

For Gaussian class-conditional distributions

$$f_k \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \qquad k = 1, \ldots, K$$

the Bayes rule (16) is called Quadratic Discriminant Analysis (QDA) and it is straightforward to see that it simplifies to

$$\delta_{QDA}(\boldsymbol{x}) = \arg\max_{i=1,\ldots,K} \delta_k \text{ where } \delta_k(\boldsymbol{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)'\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k) + \log\pi_k. \tag{17}$$

This discrimination function is called quadratic since the boundary between R_i and R_j, $\{\boldsymbol{x} \in \boldsymbol{\Omega} : \delta_i(\boldsymbol{x}) = \delta_j(\boldsymbol{x})\}$ is a quadratic function. The middle term in (17) is the well known Mahalanobis distance between $\boldsymbol{x}$ and $\boldsymbol{\mu}_k$.

Usually the parameters $\boldsymbol{\mu}_k$, π_k and $\boldsymbol{\Sigma}_k$ are unknown and have to be estimated from the sample, e.g. with the maximum likelihood estimators (18), (19) and (20).

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \boldsymbol{X}_{ik}, \qquad k = 1, \ldots, K \tag{18}$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{2(n-1)} \sum_{k=1}^{K} \sum_{i=1}^{n} (\boldsymbol{X}_{ik} - \hat{\boldsymbol{\mu}}_k)'(\boldsymbol{X}_{ik} - \hat{\boldsymbol{\mu}}_k), \qquad k = 1, \ldots, K \tag{19}$$

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad k = 1, \ldots, K \tag{20}$$

Plugging in the ML estimates $\hat{\boldsymbol{\mu}}_k$ for $\boldsymbol{\mu}_k$, $\hat{\pi}_k$ for π_k and $\hat{\boldsymbol{\Sigma}}_k$ for $\boldsymbol{\Sigma}_k$ yields the decision rule $\hat{\delta}_{QDA}$.

2.2 Linear Discriminant Analysis

For normal populations with the same covariance matrix

$$f_k \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}), \qquad k = 1, \ldots, K$$

the Bayes rule (16) is called Linear Discriminant Analysis δ_{LDA} and simplifies to

$$\delta(\boldsymbol{x})_{LDA} = \arg \max_{i=1,\ldots,K} \delta_k \text{ where } \delta_k(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log \pi_k. \tag{21}$$

Clearly this discrimination rule is of the form (8) and can thus be called linear. Plugging in the ML estimates for $\boldsymbol{\mu}_k$ (18) and using the pooled covariance estimator

$$\hat{\boldsymbol{\Sigma}}_{pooled} = \frac{1}{2(n-1)} \sum_{k=1}^{K} \sum_{i=1}^{n_k} (\boldsymbol{X}_{ik} - \hat{\boldsymbol{\mu}}_k)'(\boldsymbol{X}_{ik} - \hat{\boldsymbol{\mu}}_k) \tag{22}$$

yields the so called Fisher rule. For the two-class case, LDA will simplify to

$$\delta_F(\boldsymbol{x}) := \begin{Bmatrix} 1 & \text{if } (\boldsymbol{x} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) > \log(\frac{L(1,2)}{L(2,1)} \frac{\pi_2}{\pi_1}) \\ 0 & \text{else} \end{Bmatrix},$$

where $\boldsymbol{\mu} = (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)/2$. This case will be particularly important for the asymptotic analysis of different discrimination rules conducted in section 2.5. Plugging in the maximum likelihood estimates in (2.2) yields

$$\hat{\delta}_F(\boldsymbol{x}) := \begin{Bmatrix} 1 & \text{if } (\boldsymbol{x} - \hat{\boldsymbol{\mu}})'\hat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2) > \log(\frac{L(1,2)}{L(2,1)} \frac{\hat{\pi}_2}{\hat{\pi}_1}) \\ 0 & \text{else} \end{Bmatrix}.$$

For reasons of notational simplicity we will henceforth assume 0-1 loss (12) of misclassification and equal prior probabilities $\pi_k = 1/K, k = 1, \ldots, K$ such that

$$\log(\frac{L(i,j)}{L(j,i)} \frac{\pi_j}{\pi_i}) = 0$$

2.3 Regularized Discriminant Analysis

The application of QDA requires an enormous number of coefficients to be estimated, depending on the number of classes k and the number of features p. If $n_k < p$ the application of QDA becomes impossible since the sample covariance matrix $\mathbf{\Sigma}_k$ will not have full rank and can therefore not be inverted. When n_k is close to p but still $n_k > p$ the quality of the estimates clearly suffers in the sense that the variance of the estimates will increase rapidly. This problem led Friedman (1988) to introduce the regularized discriminant analysis, characterized by the covariance matrix

$$\hat{\mathbf{\Sigma}}_k(\alpha) = \alpha\hat{\mathbf{\Sigma}}_k + (1-\alpha)\hat{\mathbf{\Sigma}}_{pooled}. \tag{23}$$

This can be interpreted as an attempt to reduce the variance of the estimations by accepting a bias in the estimation of the covariance matrices. The level of bias is controlled by the parameter α which will be determined by means of cross validation. The same reasoning clearly applies to the case where p is close to n for the estimation of $\mathbf{\Sigma}_{pooled}$. The bias-variance tradeoff will play an even more critical role in high-dimensional feature spaces, as it will be discussed in section 2.5.

The effect of low sample sizes can be best illustrated by considering the sample covariance matrix of class k, $\mathbf{\Sigma}_k$, as a mapping in $\mathbb{R}^p$ and looking at its spectral decomposition

$$\mathbf{\Sigma}_k = \sum_{i=1}^{p} e_{ik} v_{ik} v_{ik}'$$

where e_{ik} is the ith eigenvalue of $\mathbf{\Sigma}_k$ in decreasing order and v_{ik} is the corresponding eigenvector. The inverse has the following representation

$$\mathbf{\Sigma}_k^{-1} = \sum_{i=1}^{p} \frac{1}{e_{ik}} v_{ik} v_{ik}'.$$

After multiplying with -2 the discrimination score of (17) becomes

$$\delta_k(\boldsymbol{x}) = \sum_{i=1}^{p} \frac{1}{e_{ik}} (v_{ik}'(\boldsymbol{x} - \boldsymbol{\mu}_k))^2 + \sum_{i=1}^{p} \log e_{ik} - 2\log \pi_k$$

and is highly dependent on the values of the smallest eigenvalues. As the sample size n_k decreases the largest eigenvalue tends to increase and the lowest eigenvalue tends to decrease. The condition of inverting the sample covariance matrix gradually decreases. When $n_k < p$ the covariance matrix loses a rank and the smallest eigenvalue is 0. This then becomes an ill conditioned problem.

The condition number of a function, the covariance matrix, is seen as a linear function on $\mathcal{R}^p \to \mathcal{R}^p$, is the rate of change at which the solution will change with respect to a change in the argument. Since the covariance matrix $\mathbf{\Sigma}$ is normal, in l_2 the condition number is

$$\kappa(\mathbf{\Sigma}) = \left| \frac{\lambda_{\max}(\mathbf{\Sigma})}{\lambda_{\min}(\mathbf{\Sigma})} \right|.$$

2.4 Comparing Different Discrimination Rules

The misclassification error is a natural measure of the quality of a classification problem and was defined in (13). However, for the asymptotic analysis a simple expression is required. Therefore,

only the two-class case $k = 2$ and only the posterior misclassification error of observations of class $\mathcal{C}_1$ will be considered, defined by

$$W(\delta, \theta) = P_\theta[\delta(X) < 0 | \mathcal{C}_1], \tag{24}$$

where θ is a parameter from a parameter space Γ yet to be defined. The worst case posterior classification error is

$$W(\delta) = \max_{\theta \in \Gamma} W(\delta, \theta). \tag{25}$$

The misclassification rates of different discrimination functions are bounded. In the two-class case the lower bound is random guessing. If the parameters $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k are known, the upper bound is the Bayes rule, which is the best discriminator for a given loss function. If the parameters are unknown and are substituted by its ML estimates, yielding the Fisher rule, it is at least asymptotically optimal. Thus, for a fixed $p < n$ it is known that

$$W(\delta_F) \xrightarrow{n \to \infty} 1 - \Phi(c/2)$$

the misclassification rate converges to the Bayes risk, where c is a measure of the signal strenght, as defined in (26).

2.5 Discriminant Analysis in High-Dimensional Feature Spaces

Let us consider a high-dimensional feature space $p >> n$. For reasoning in the finite case we will just assume that the number of features is of the order 10^4, e.g. 20000, and the number of observations is of the order 10^2, e.g. 150. For asymptotic reasoning the dimension of the feature space p is always modeled through its dependence on n, $p_n \to \infty$, and can be characterized as $n = o(p_n)$. The index will be dropped from now on.

Two problems arise in this setting which are closely related. First, the application of discriminant methods LDA, QDA or RDA is not possible since $\hat{\boldsymbol{\Sigma}}$ is singular and can therefore not be inverted, and secondly even if we fix this problem the variance of the estimation is enormous. To shed light on these issues Bickel and Levina (2004) have shown, that even if $\hat{\boldsymbol{\Sigma}}$ is replaced by the Moore-Penrose pseudo inverse $\hat{\boldsymbol{\Sigma}}^-$ the worst case misclassification error of the Fisher discriminant function (2.2) is no better than random guessing.

Theorem 1. *Consider the parameter space*

$$\Gamma = \{(\mu_0, \mu_1, \boldsymbol{\Sigma}) : \underbrace{(\mu_1 - \mu_2)' \boldsymbol{\Sigma}^{-1} (\mu_1 - \mu_2) \geq c^2}_{(A1)}, \underbrace{k_1 \leq \lambda_{min}(\boldsymbol{\Sigma}) \geq \lambda_{max}(\boldsymbol{\Sigma}) \geq k_2}_{(A2)}, \underbrace{\mu_i \in B, i = 0, 1}_{(A3)}\} \tag{26}$$

(A_1) is a condition on the signal strength of the discrimination problem. The Mahalanobis distance between the population centers of groups $\mathcal{C}_1$ and $\mathcal{C}_2$ has to been sufficiently large for the discrimination between two populations to be feasible. Condition (A_2) guarantees that both $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}^{-1}$ are not ill-conditioned since $\kappa(\boldsymbol{\Sigma}) < k_2/k_1$. Condition $(A3)$ is a technical condition, where $B = \{\mu \in l_2 : \sum_{j=1}^{\infty} a_j \mu_j^2 \leq d^2\}$ and $a_j \to \infty$ is a compact subset of l_2.

If $n = o(p)$ then

$$W_\gamma(\delta_F) \to \frac{1}{2}.$$

As p diverges to infinity the spectrum of the covariance matrix, which is a $p \times p$ matrix, will diverge and so will the condition number of the covariance matrix. When $p \to \infty$ the

misclassification rate of the Fisher rule converges to 1/2, which is equivalent to random guessing. Thus the bad performance of the Fisher rule is due to the numerical instability of the estimation of the covariance matrix as the dimension of the feature space diverges.

In order to solve this problem some form of regularization will be necessary. Several possibilities were suggested to overcome this problem. A popular approach is the assumption of independent features within classes yielding an LDA with diagonal covariance matrix, often referred to as Independence rule. This will dramatically reduce the number of coefficients to be estimated and thereby often improve the performance of the classifier. This classifier will be introduced and studied in section 2.6. Under the (reasonable) assumption that not all features contribute to classification this rule can further be regularized when it is preceded by a feature selection. This method called Feature Annealed Independence Rule will be introduced in section 2.7. Another approach that solves both the problems of inverting the covariance matrix as well as the numerical is a version of Regularized Discriminant Analysis introduced in section 2.9.

2.6 Independence Rule

The Independence Rule (IR) is obtained when the assumption that features are independent are added to the assumptions made for LDA. This yields the following discriminant function

$$\delta_{IR}(\boldsymbol{x}) := \arg\max_{k=1,\dots,K} \delta_k \text{ where } \delta_k = \sum_{j=1}^{p} \frac{x_j - \mu_{jk}}{\sigma_j^2} + 2\log(\pi_k), \tag{27}$$

where the parameters $\boldsymbol{\mu}_k$, and $\boldsymbol{\sigma}$ have to be estimated from the sample, e.g. with the maximum likelihood estimators. This discrimination rule allows a simple interpretation. If we call $\boldsymbol{\mu}_k$ the centroid of class k, then the IR will classify an observation as belonging to class k if $\boldsymbol{x}$ is closest to this centroid, in the sense of the standardized squared distance corrected by an expression containing the prior probabilities. For the two-class case the IR reduces to

$$\delta_{IR}(\boldsymbol{x}) = \begin{Bmatrix} 1 & \text{if } (\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{D}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) > \frac{\log \pi_1}{\log \pi_2} \\ 2 & \text{else} \end{Bmatrix}$$

where $\boldsymbol{D} = \text{diag}(\boldsymbol{\Sigma})$. Bickel and Levina (2004) have studied the consequences of making the evidently invalid assumption of independent covariates in the following theorem.

Theorem 2. *If* $\log(p) = o(n)$*, then*

$$\limsup_{n\to\infty} W(\delta_{IR}) = 1 - \Phi(\frac{\sqrt{K_0}}{1+K_0}c)$$

where $K_0 = \max_\Gamma \frac{\lambda_{\max}(\boldsymbol{\Sigma}_0)}{\lambda_{\min}(\boldsymbol{\Sigma}_0)}$ *is the worst case condition number of* $\boldsymbol{\Sigma}_0$*, the correlation matrix* $\boldsymbol{\Sigma}_0 = \boldsymbol{D}^{-1/2}\boldsymbol{\Sigma}\boldsymbol{D}^{-1/2}$*.*

If the assumption of independent features is correct and $\boldsymbol{\Sigma} = c\boldsymbol{I}_p$ is a multiple of the identity then $K_0 = 1$ and the misclassification rate converges to the Bayes level. However, if the condition number $\kappa(\boldsymbol{\Sigma}_0)$ diverges, this is the case when either $\lambda_{\min} \to 0$ or $\lambda_{\max} \to \infty$, then also the IR is reduced to random guessing. In many situations, however, the independence rule will outperform the Fisher rule. A small comparison between the Fisher rule and the IR (for p approaching n) was conducted by means of a simulation study in Tibshirani et al. (2003).

The IR, however, has two weaknesses. Firstly, since all features are used for prediction, the interpretability of the model is very limited. Secondly and more importantly, the prediction

quality can be negatively affected by the large number number of predictors due to noise accumulation. Noise accumulation describes the phenomenon that a large number of noise-only-features can deteriorate the estimations e.g. of the population mean and therefore reduce the quality of prediction. In fact, Fan and Fan (2008) demonstrate that even for the IR classification using all the features can be as poor as random guessing.

Theorem 3. *Consider the parameter space*

$$\Gamma = \{(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) : (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{D}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \geq C_p, \lambda_{max}(\boldsymbol{R}) \leq b_0, \min_{1 \leq j \leq p, k=1,2} \sigma_{kj}^2 > 0\}$$

where $\boldsymbol{R} = \boldsymbol{D}^{-1/2} \boldsymbol{\Sigma} \boldsymbol{D}^{-1/2}$ *is the correlation matrix,* C_p *is a deterministic positive sequence, denoting the lower bound for the overall signal,* λ_{max} *is the maximum eigenvector of the correlation matrix* $\boldsymbol{R}$*,* b_0 *is a positive constant.*

Further, suppose that $\log(p) = o(n)$ *and* $n = o(p)$*. These assumptions characterize the asymptotic behavior of* n *and* p_n*, where* p *dominates* n *but is dominated by* e^n*. This relation between* n *and* p *is called high-dimensional.*

Furthermore assume that $p/(nC_p) \to 0$ *and* $\{\frac{n_1 n_2}{pn}\}^{1/2} C_p \to 0$ *which characterizes the trade-off between signal strength and dimensionality under which the following relation holds.*

Then for the worst case classification error $W(\delta)$ *the following relation holds*

$$W(\hat{\delta}_{IR}) \xrightarrow{P} \frac{1}{2}.$$

To overcome both the problem of limited interpretability and noise accumulation feature selection seems like a most obvious thing to do. Several variants of feature selection exist in the literature, which can be categorized in soft and hard thresholding.

2.7 Feature Annealed Independence Rule

Fan and Fan (2008) suggest to select a subset of features yielding the Features Annealed Independence Rule (FAIR) by means of applying a component-wise two sample t-test, testing the following hypothesis

$$H_{j0} : \exists k, l : \mu_{jk} \neq \mu_{jl} \text{ versus } H_{j1} : \mu_{j1} = \cdots = \mu_{jK} \qquad j = 1, \ldots, p, \tag{28}$$

for two classes. The multi-class case can be tested by means of a one-way ANOVA. This is a very popular method for selecting features for classification in the field of genomics. The resulting method can be considered a two-step experiment, where the first step consists of selecting the m most significant features according to component wise two-sample t-tests and the second step is simply applying the IR to theses m features. Consider the two-sample t-test statistic

$$T_j = \frac{\hat{\mu}_{kj} - \hat{\mu}_{lj}}{\sqrt{\hat{\sigma}_{1j}^2 / n_k + \hat{\sigma}_{2j}^2 / n_l}} \qquad j = 1, \ldots, p, 1 \leq k, l \leq K. \tag{29}$$

The FAIR discrimination function is derived when only those features are used for classification that qualify in the sense that their absolute t-statistic is greater than a critical value b, yielding

$$\hat{\delta}_{FAIR}(\boldsymbol{x}) = \arg \max_{k=1,\ldots,K} \delta_k \text{ where } \delta_k = \sum_{j=1}^{p} \frac{x_j - \hat{\mu}_j}{\hat{\sigma}_j^2} \cdot \mathbf{1}_{\{\sqrt{n/(n_1+n_2)}|T_j| > b\}} \tag{30}$$

where T_j denotes the two-sample t-test (29).

This method can be seen as a hard-thresholding feature selection method since the feature j is either included or not depending on $\sqrt{n/(n_1+n_2)}|T_j| > b$.

Fan and Fan (2008) further show that this feature selection method fulfills a sure-screening property. Consider the problem of selecting features for the purpose of discriminating between two-classes by means of a t-statistic.

Theorem 4. *Under the assumption that*

- n_1 *and* n_2 *are fairly equal, that is,* $c_1 \leq n_1/n_2 \leq c_2$ *with positive constants* c_1 *and* c_2*,*
- *the vector* $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ *is sparse, that is most of its components are 0 and without loss of generality only the first* s *are non-zero,*
- ϵ_{ijk} *and* ϵ_{ijk}^2 *satisfy Cramer's condition, that is, there exist constant* ν_1 *and* ν_2*,* M_1 *and* M_2*, such that* $E|\epsilon_{ijk}|^m \leq m! M_1^{m-2}\nu_1/2$ *and* $E|\epsilon_{ijk}^2 - \sigma_{jk}^2| \leq m! M_2^{m-2}\nu_2/2$ *for all* $m = 1, 2, \ldots$*, and that*
- *the diagonal elements of both* $\boldsymbol{\Sigma}_1$ *and* $\boldsymbol{\Sigma}_2$ *are bounded away from 0*

then

$$P(\min_{j \leq s} |T_j| \geq x \text{ and } \max_{j > s} |T_j| < x) \to 1. \tag{31}$$

This theorem states that as long as the signal level is sufficiently high, the two sample t-test can at least asymptotically select the signal features with probability 1.

The question how to chose the number of features still remains. There are several possibilities to choose the number of features for the final model. First one can add all those features with a significant t-test at a, say, 0.05 level. This approach, however, requires the assumption of Gaussian features. Fan and Fan (2008) choose a different approach, which relaxes the distributional assumption to some extent and give expression (32) that delivers the optimal number of features. One weakness of the approach used in FAIR is that the estimation of the largest eigenvalue becomes more and more inaccurate as the dimension of the feature space increases. A non-parametric approach would be to obtain the optimal number of features by means of a cross validation.

$$\hat{m}_1 = \arg \max_{1 \leq m \leq p} \frac{1}{\hat{\lambda}_{max}^m} \frac{n[\sum_{j=1}^m T_j^2 + m(n_1 - n_2)/n]^2}{m n_1 n_2 + n_1 n_2 \sum_{j=1}^m T_m^2} \tag{32}$$

One drawback of hard-thresholding is its 'jumpy' nature. Tibshirani et al. (2002) and Tibshirani et al. (2003) have investigated by means of a simulation study the difference between soft and hard-threshold feature selection and have introduced the Nearest shrunken centroids method. Intuitively this can be interpreted as gradually diminishing the influence of features as the threshold increases rather that leaving it unchanged until a certain value of the threshold has been reached and then kicking it out completely.

2.8 Nearest Shrunken Centroids

Like FAIR, the classification method called *Nearest Shrunken Centroids* (NSC) is a variant of the IR and will attempt to effectively eliminate most non-contributing genes leaving only those that are significant for classification. However, it will use a soft-threshold method for feature

selection. Fan and Lv (2008) show that this method is a member of the family of penalized quasi-likelihood methods since it can be seen as a ridge regression with ridge parameters tending to ∞. The geometrical intuition is that for classification purposes a denoised version of class centroids, called shrunken centroids, are used. The amount of shrinkage is continuos and determined by cross validation.

The shrinkage procedure consists of the following steps. Calculate

$$d_{jk} = \frac{\hat{\mu}_{jk} - \hat{\mu}_j}{m_k \cdot (\hat{\sigma}_i + \hat{\sigma}_0)} \tag{33}$$

where $\hat{\mu}_i$ is the overall mean, $\hat{\mu}_{ik}$ the class-wise mean of gene i, $\hat{\sigma}_i$ is the pooled within-class standard deviation for gene i. $m_k = \sqrt{1/n_k - 1/n}$ corrects the expression such that the denominator is equal to the standard error of the numerator, such that d_{ik} has unit variance. $\hat{\sigma}_0$ is a small constant such that large d_{jk} values do not arise from small nominators.

Soft thresholding is used to shrink theses values towards zero by

$$d'_{jk} = \text{sign}(d_{jk})(|d_{kj}| - \Delta)_+$$

where Δ is obtained by means of cross validation. The centroids of genes with a small value d_{jk} will be shrunken towards the overall centroid such that they do not contribute any more for classification. The shrunken centroids are obtained by using (33)

$$\hat{\mu}'_{jk} = \hat{\mu}_j + m_k \cdot (\hat{\sigma}_i + \hat{\sigma}_0) \cdot d'_{jk}. \tag{34}$$

The value of Δ that minimized prediction error will be identified and all those genes with a value $d_{jk} < \Delta$ will be excluded from the feature set used for prediction, since in that case $\hat{\mu}'_{jk} = \hat{\mu}_j$ the new class centroid will equal the overall centroid. One can also see that in the NSC method, the group centroids of each gene are shrunken individually. This is based on the assumption that genes are independent of each other. This yields the discrimination rule

$$\delta_{NSC}(\boldsymbol{x}) = \arg \min_{k=1,\ldots,K} \delta_k \text{ where } \delta_k = -\sum_{i=1}^{p} \frac{(x_{ik} - \mu'_{jk})}{\sigma_j^2} + 2\log(\pi_k) \tag{35}$$

where μ'_{jk} are the shrunken centroids defined in (37).

In Figure 1 one can see the effect of an increase in the threshold Δ on the number of features and on the classification error. The lowest misclassification error is obtained at a threshold of about 5. At that value only seven genes are used for classification. The centroids are depicted in Figure 2 where the value of each of the seven genes is illustrated for centroid one (red) and centroid two (green).

2.9 Shrunken Centroids Regularized Linear Discriminant Analysis

This method builds upon the idea of RDA by Friedman (1988) as introduced in section 2.3, however it was developed for high dimensional feature spaces. A detailed introduction can be found in Guo et al. (2005).

It uses a different approach to solving the problem of inverting the singular covariance matrix than for instance IR, FAIR or NSC. The idea is to regularize it by adding a diagonal matrix, thus

$$\tilde{\boldsymbol{\Sigma}} = \alpha\hat{\boldsymbol{\Sigma}} + (1 - \alpha)\boldsymbol{I}_p. \tag{36}$$

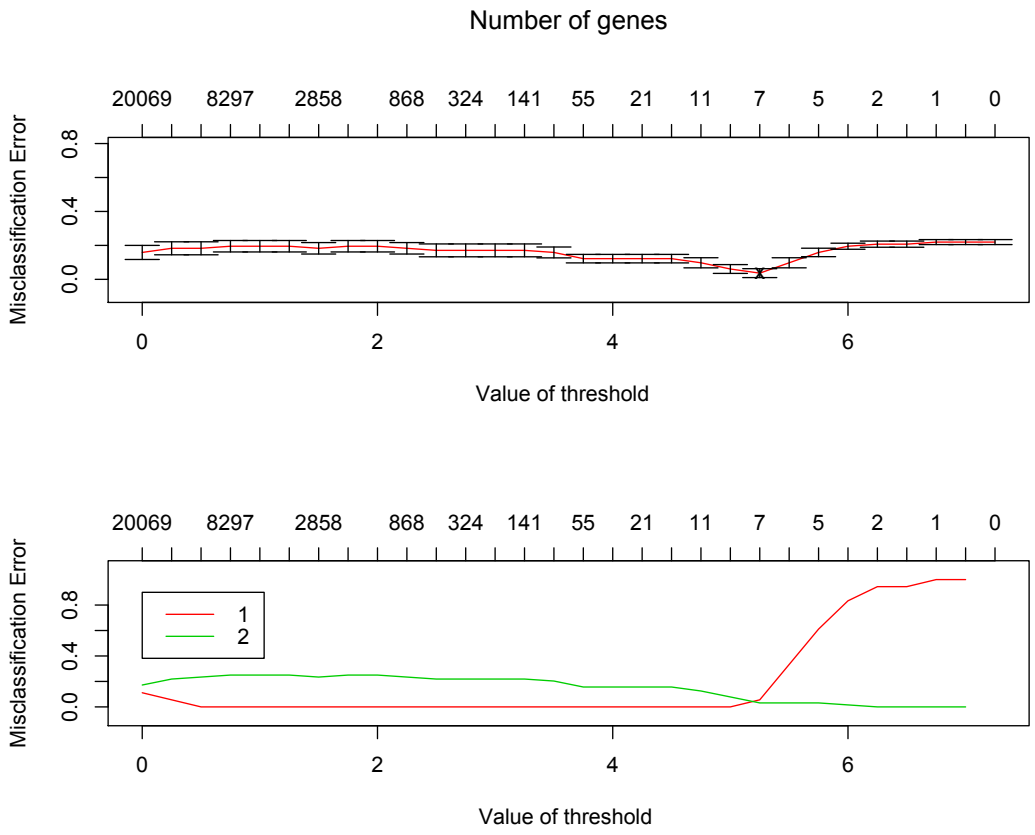

Figure 1: Illustration of the effect of increasing the threshold on the number of features and on the misclassification rate.

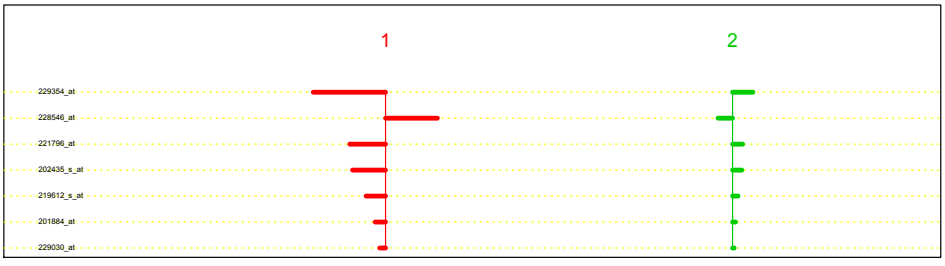

Figure 2: An illustration of centroids 1 and 2 at the optimal threshold level.

where α is a regularization parameter obtained by means of cross validation. By introducing a slightly biased covariance estimate, not only is the singularity problem resolved, since clearly $\tilde{\boldsymbol{\Sigma}}$ will have full rank, but also the sample covariance estimate is stabilized. This is a classical bias variance tradeoff. Additionally, the idea of shrinking the centroids in order to denoise them, as for NSC in section 2.8, was incorporated using

$$\hat{\boldsymbol{\mu}}' = \operatorname{sgn}(\hat{\boldsymbol{\mu}})(|\hat{\boldsymbol{\mu}}| - \Delta)_+, \tag{37}$$

where Δ is a shrinking parameter. Therefore the parameter space of the SCRDA method is two-dimensional $\{(\alpha, \Delta) : \alpha \in [0, 1], \delta \in [0, \infty)$.

The application of a LDA (21) with a regularized covariance matrix $\tilde{\boldsymbol{\Sigma}}$ defined in (36) and shrunken centroids defined in (37) yields the following discrimination function

$$\delta(\boldsymbol{x})_{SCRDA} = \arg\max_{i=1,\dots,K} \delta_k \text{ where } \delta_k(\boldsymbol{x}) = \boldsymbol{x}'\tilde{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k'\tilde{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\mu}_k + \log \pi_k. \tag{38}$$

3 Support Vector Machines

A *Support Vector Machine* (SVM) is a machine learning algorithm for solving classification problems with two classes. There are two ways to introduce SVMs. The classical approach is a geometrical one building on the ideas of *perceptrons*, hyperplanes dividing the sample space in two subsets. Bear in mind that all classification methods can be seen as method of dividing the sample space into disjoint regions, see (9), which can then be used for classification. This approach, see section 3.1, is how SVMs were introduced originally and furthermore it gives a good intuition about how the method works. However, it has been shown that the SVM can be seen as the solution to an optimization problem with a certain form of Tikhonov regularization, see section 3.2. This perspective provides a general framework, in which also *Logistic Regression* is a member, see section 4. Furthermore it allows to relate to the optimization problem (10) and thus allows further analysis.

SVMs can be extended to a classification method for multiple classes in several ways, which will be described in section 3.4.

3.1 Geometrical View of SVMs

Rosenblatt (1958) introduced the idea of using separating affine hyperplanes for classification purposes. Consider a set of observations $(\boldsymbol{x}_i, c_i)$, where $\boldsymbol{x}_i \in \mathbb{R}^p, c_i \in \{-1, 1\}$ for $i = 1, \dots, n$. An affine hyperplane can be defined by means of a vector $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)'$ through

$$\{\boldsymbol{x} : \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \beta_0 + \boldsymbol{\beta}'\boldsymbol{x} = 0\}.$$

The sign of $\beta_0 + \boldsymbol{\beta}'\boldsymbol{x}$ determines on which side of the hyperplane the observation $\boldsymbol{x}$ lies, yielding the classification rule

$$\delta(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{\beta}'\boldsymbol{x} + \beta_0). \tag{39}$$

Therefore an optimal classifying hyperplane satisfies

$$c_i(\boldsymbol{\beta}'\boldsymbol{x}_i + \beta_0) > 0 \text{ for } i = 1, \dots, n.$$

The further a point is located from the affine hyperplane the greater $\boldsymbol{\beta}'\boldsymbol{x} + \beta_0$. Classifiers of this type were called *perceptrons*. The *perceptron learning algorithm* attempts to find a separating

affine hyperplane that minimizes the distance of misclassified points to the hyperplane. This problem can be formulated as an optimization problem and solved by means of a stochastic gradient descent method.

There are, however, a number of weaknesses to this approach. First of all the solution to this problem is not necessarily unique, and second when the classes are not perfectly separable by a hyperplane the algorithm will not converge. These problems were tackled by Cortes and Vapnik (1995). They suggest to obtain the optimal separating hyperplane by maximizing the distance between any observation and the closest point of the separating border, yielding

$$\begin{aligned} &\max_{\boldsymbol{\beta},\beta_0,\|\boldsymbol{\beta}\|=1} M \\ &\text{subject to } c_i(\boldsymbol{x}_i'\boldsymbol{\beta}+\beta_0) \geq M, i = 1,\ldots,n. \end{aligned}$$

If these conditions are fulfilled, the distance of any observation $\boldsymbol{x}_i$ to the separating hyperplane will be at least M, and thus, the distance between any two observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from different classes will be at least $2M$. This approach provides a unique solution and improves the classification performance. However, it still requires perfectly separable classes. Therefore, the problem

$$c_i(\boldsymbol{x}_i'\boldsymbol{\beta}+\beta_0) \geq M$$

was modified to

$$\begin{aligned} &c_i(\boldsymbol{x}_i'\boldsymbol{\beta}+\beta_0) \geq M(1-\xi_i) \\ &\text{where } \forall i, \xi_i \geq 0, \sum_{i=1}^{n} \xi_i = \text{ const.} \end{aligned}$$

where $\xi = (\xi_1,\ldots,\xi_n)$ are the so-called slack variables that were introduced in order to allow for overlapping classes. The slack variable ξ_i captures the proportional amount by which the prediction $\boldsymbol{x}_i'\boldsymbol{\beta}+\beta_0$ is on the wrong side of the separating hyperplane. Therefore, by bounding $\sum_{i=1}^{n}\xi_i$ the total proportional amount by which predictions falls on the wrong side of the margin is bounded. Thus, maximizing

$$\begin{aligned} &\min \|\beta\| \\ &\text{subject to } \forall i, c_i(\boldsymbol{x}_i'\boldsymbol{\beta}+\beta_0) \geq (1-\xi_i) \\ &\text{and } \xi_i \geq 0, \sum \xi_i \leq \text{ const.} \end{aligned}$$

yields a separating hyperplane for observations that are not perfectly separable. This approach is called Support Vector Machine (SVM) and has the equivalent expression

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to } \xi_i \geq 0, c_i(\boldsymbol{\beta}'\boldsymbol{x}_i+\beta_0) \geq 1-\xi_i, i = 1,\ldots,n \end{aligned} \tag{40}$$

where C is referred to as the cost parameter. This is a quadratic problem with linear inequality constraints, hence it is a convex optimization problem, and can, thus, be solved with standard algorithms.

3.2 SVMs as a Solution to a Tikhonov Regularized Optimization Problem

The SVM can be stated as special version of the regularized learning problem

$$\min_{f\in\mathcal{H}} \frac{1}{l}\sum_{i=1}^{n} \xi(f(\boldsymbol{x}_i), c_i) + \lambda\|f\|_K^2, \tag{41}$$

where $\|f\|_K^2$ is the norm of any function f, e.g. $f(\boldsymbol{x}) = \boldsymbol{\beta}'\boldsymbol{x}+\beta_0$, in a *Reproducing Kernel Hilbert Space* $\mathcal{H}$, defined by a positive definite kernel function K, see e.g. Hastie et al. (2009). The loss function, denoted by $\xi()$, indicates the penalty that has to be payed, when a classification $f(\boldsymbol{x}_i)$ differs from the actual class c_i membership. λ is a regularization parameter balancing the trade-off between prediction accuracy and a function with a small norm in $\mathcal{H}$. The most natural loss function is the 0-1 loss function

$$\xi(f(\boldsymbol{x}), c) = \begin{Bmatrix} 0 & \text{if sign}f(\boldsymbol{x}) = c \\ 1 & \text{otherwise} \end{Bmatrix},$$

which is of course the well known misclassification rate (14). However, this leads to a non-convex problem, so instead a continuos upper bound, called the *hinge loss function*,

$$\xi_{L1}(f(\boldsymbol{x}), c) = (1 - cf(\boldsymbol{x}))_+, \tag{42}$$

where $(x)_+ = \max(k, 0)$ will be used. It can be shown that applying the *hinge loss function* and setting

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{\beta}\cdot\boldsymbol{x} + \beta)$$

and $C = \frac{1}{2\lambda l}$ yields the classical SVM (40). It cannot be emphasized enough, that the support vector machine, which was originally motivated in a purely geometric fashion, is, thus, naturally regularized!

The use of (42) can easily be justified. It will give zero loss for $c_i = 1$ if $f(\boldsymbol{x})$ is large and positive and for $c_i = -1$ if $f(\boldsymbol{x})$ is large and negative. If the signs of c_i and $f(\boldsymbol{x}_i)$ do not match, then it will induce a loss.

This optimization problem, (41), relates to (10) in the sense that it adds a regularization term to the (empirical version of) (10), if a certain loss function is used. The loss function was also generalized $L(\text{sign}(f(\boldsymbol{x})), c) = \xi(f(\boldsymbol{x}), c)$ such that is allows for classifiers with a support $\mathbb{R}$. This is of course only possible for the two-class case.

Different loss functions induce different classification methods, e.g.

$$\xi_{log}(f(\boldsymbol{x}), c) = \log(1 + \exp^{-cf(\boldsymbol{x})}) \tag{43}$$

$$\xi_{L2}(f(\boldsymbol{x}), c) = (1 - cf(\boldsymbol{x}))_+^2 \tag{44}$$

where the use of ξ_{log} yields the L1- regularized logistic regression model, introduced in section 4, and ξ_{L2} yields the L2 regularized SVM.

3.3 Support Vector Machines in High-Dimensional Feature SSpaces

As we have seen, SVMs are naturally regularized and are therefore well suited for high dimensional feature spaces. However, two other adaptions are typical.

First, the use of *linear Support Vector Machines*. SVMs usually map the observations into a higher-dimensional space Z using non-linear functions $\boldsymbol{z}()$. This allows to separate observations

of two classes, which cannot be separated in the original space. This approach is referred to as a *non-linear SVM*. In applications with already high-dimensional feature spaces SVMs with $p > n$ classes can always be separated in the original feature space and the mapping becomes obsolete. This yields the so-called *linear SVM*.

Another problem in feature spaces with many features and relatively few observations is noise-fitting. This issue is usually resolved by effective regularization. Standard regularization terms are $\|\cdot\|_1$ or the $\|\cdot\|_2$, which can also be seen as special versions of Tikhonov regularization. For reasons of interpretability, sparse models are preferred leading to the use of the norm $\|\cdot\|_1$. This yields the following optimization problem

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_1 + C \sum_{i=1}^{n} \xi(\boldsymbol{\beta}; \boldsymbol{x}_i, c_i) \tag{45}$$

where $\xi()$ is a non-negative convex loss function. The norm $\|\cdot\|_1$ is supposed to avoid fitting to noise and the parameter C is supposed to balance between the loss and regularization. L1 regularization is known to produce sparse predictor vectors $\boldsymbol{\beta}$ and therefore distinguish between important and unimportant features. This comes with the drawback that it renders the target function indifferentiable and, thus, makes optimization more complex.

3.4 Multi-Class Classification

Several approaches to extend the SVM from a binary classification method to a classification method for $K > 2$ classes have been suggested. An obvious approaches is the *one-vs-one* classifier computing one SVM for every pair of classes. The observation is classified to the class winning the most pairwise contests. Another much used approach is the *one-vs-all* method. For every class k the distance to the separating affine hyperplane is calculated indicating a level of confidence for group k. The observation is then accredited to that class with the highest confidence level.

Furthermore there are also more sophisticated multi-class support vector machines such as the version developed by Crammer and Singer (2001). In this paper they generalize the optimization problem to allow for more than two classes. Unfortunately however, the regularization used is a L2 regularization, which does not yield sparse models.

4 Logistic Regression

Logistic regression is a member of the family of *generalized linear models*. It arises from the approach to model the posterior probability of an event as a function of a set of covariates. Assume for now, there are two classes, $\mathcal{C}_1$ and $\mathcal{C}_2$. Since the support of probabilities is $[0, 1]$, it is problematic to directly model the posterior probability of an event as a linear function of a set of observations such as $\beta_0 + \boldsymbol{\beta}'\boldsymbol{x}$. Therefore the log-odds of an event are modeled as a linear function of a set of covariates. The odds are defined as the posterior probability of a case divided by the probability of a non case. The support of the odds is $[0, \infty]$. Taking the logarithm extends the support to $[-\infty, \infty]$, which is more appropriate for a linear function. Thus the model has the form

$$\log(\frac{P[\mathcal{C} = 1|X = \boldsymbol{x}]}{P[\mathcal{C} = 2|X = \boldsymbol{x}]}) = \log(\frac{P[\mathcal{C} = 1|X = \boldsymbol{x}]}{1 - P[\mathcal{C} = 1|X = \boldsymbol{x}]}) = \beta_0 + \boldsymbol{\beta}'\boldsymbol{x}. \tag{46}$$

This allows to express the posterior probability as

$$P(\mathcal{C}=1|X=\boldsymbol{x})=\frac{\exp(\beta_0+\boldsymbol{\beta}'\boldsymbol{x})}{\exp(\beta_0+\boldsymbol{\beta}'\boldsymbol{x})+1}=\frac{1}{\exp(-\beta_0+\boldsymbol{\beta}'\boldsymbol{x})+1}. \tag{47}$$

This approach can be extended to the K-class case, where the set of equations

$$\begin{aligned}
\log\frac{P[\mathcal{C}=1|X=\boldsymbol{x}]}{P[\mathcal{C}=K|X=\boldsymbol{x}]}&=\beta_{10}+\boldsymbol{\beta}_1'\boldsymbol{x}\\
\log\frac{P[\mathcal{C}=2|X=\boldsymbol{x}]}{P[\mathcal{C}=K|X=\boldsymbol{x}]}&=\beta_{20}+\boldsymbol{\beta}_2'\boldsymbol{x}\\
&\vdots\\
\log\frac{P[\mathcal{C}=K-1|X=\boldsymbol{x}]}{P[\mathcal{C}=K|X=\boldsymbol{x}]}&=\beta_{(K-1)0}+\boldsymbol{\beta}_{K-1}'\boldsymbol{x}
\end{aligned} \tag{48}$$

extend (46) and the set of equations

$$P[\mathcal{C}=k|X=\boldsymbol{x}]=\frac{\exp(\beta_{k0}+\boldsymbol{\beta}_k'\boldsymbol{x})}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\boldsymbol{\beta}_l'\boldsymbol{x}),\quad k=1,\ldots,K-1} \tag{49}$$

$$P[\mathcal{C}=K|X=\boldsymbol{x}]=\frac{1}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\boldsymbol{\beta}_l'\boldsymbol{x})} \tag{50}$$

extend (47).

The logistic regression model is usually fit by maximizing the log-likelihood, where the class memberships follow a multinomial distribution,

$$\max_{\{\beta_{0k},\boldsymbol{\beta}_k\}_1^K} l(\boldsymbol{x}_i,c_i;\theta)=\sum_{i=1}^{n}\log(P[\mathcal{C}=c_i|X=\boldsymbol{x}_i;\theta]),$$

the solution of which is obtained by the Newton-Raphson algorithm.

4.1 Comparing Linear Discriminant Analysis and Logistic Regression

Recall that LDA was also based on modeling the posterior probability of an event. It is thus possible to express the log-odds ratio

$$\begin{aligned}
\log(\frac{P[\mathcal{C}=k|X=\boldsymbol{x}]}{P[\mathcal{C}=K|X=\boldsymbol{x}]})&=\log\frac{\pi_k}{\pi_K}-\frac{1}{2}(\mu_k+\mu_K)'\boldsymbol{\Sigma}^{-1}(\mu_k-\mu_K)+\boldsymbol{x}'\boldsymbol{\Sigma}^{-1}(\mu_k-\mu_K)=\\
&=\alpha_{k0}+\boldsymbol{\alpha}_K'\boldsymbol{x}.
\end{aligned}$$

This is exactly the same modeling structure as the LR model, see (49). LDA and LR, however, differ in the way how these parameters are estimated. The LDA estimates the parameters by maximizing the full log-likelihood using the joint density

$$P[X,G=k]=\phi(X,\mu_k,\boldsymbol{\Sigma})\pi_k$$

based on the assumption that the class-specific distributions are Gaussian with mutual covariance but different means. Under this distributional assumption all that needs to be estimated are the parameters of the Gaussian distribution.

The LR splits the joint density into the marginal density of X and the conditional density

$$P[X, G = k] = P[X]P[\mathcal{C} = k|X]$$

and fits the parameters by maximizing the conditional density (50). This can be interpreted as a completely non-parametric estimation of the marginal density of X. For further details see e.g. Hastie et al. (2009). If the class-specific distributions are in fact Gaussian then the estimation is very efficient and will require about 30 % less observations than the estimation without any distributional assumptions. However, even when the class-specific distributions are not Gaussian and this assumption causes a bias, this approach can still deliver good results due to the small variance. Especially in a situation with few observations or many features. Again, this is a classic bias-variance trade-off.

4.2 Logistic Regression in High-Dimensional Feature Spaces

Logistic regression can be modified for the $p >> n$ case by including a regularizing term. In order to obtain sparse models, L1 regularization is the standard choice. The regularization can be implemented by maximizing the penalized log-likelihood

$$\max_{\{\beta_{0k}, \boldsymbol{\beta}_k\}_1^K} \left[\sum_{i=1}^{n} \log(P[\mathcal{C} = c_i|X = \boldsymbol{x}_i; \theta]) - \lambda \sum_{i=1}^{K} \|\beta_k\|_1 \right]$$

This is a convex optimization problem and can be solved e.g. by a Newton algorithm. The logistic regression has been motivated as the solution to an optimization problem including Tikhonov regularization. The regularization path, which illustrates the effect of an increasing parameter λ on the number and the coefficients of the selected features can be found in Figure 3.

5 Implementations

This section briefly comments on the implementations of the methods introduced in sections 2, 3 and 4. For LR and the SVM a myriad of different implementations exist in R, both regularized and unregularized. `LiblineaR` offers a solver for the primal L1-regularized SVM and the L1-regularized LR. The optimization algorithm used is a coordinate descent method developed by Fan et al. (2008) extending Chang et al. (2008). The convincing arguments were its incredible speed and good documentation. For the multi-class SVM of Crammer and Singer (2001) the package `LiblineaR` was used. Unfortunately this methods implemented a L2-regularization and, thus, does not deliver sparse models. To my knowledge there exists no L1-regularized implementation of a multi-class SVM to date.

The `pamr` package by Hastie et al. (2011) provides the NSC classifier, whereas the SCRDA function is provided by the `rda` package of Guo et al. (2012). These methods require an internal cross validation to obtain the optimal parameters, a shrinkage parameter for the NSC and both a shrinkage and a regularization parameter for the SCRDA method. The optimal parameters were obtained by an internal cross validation using the *minmin*-rule, which means:

1. Find all parameters/parameter sets that correspond to the minimal cross-validation error.
2. Second, select the parameter/parameter set that use the minimal number of features.

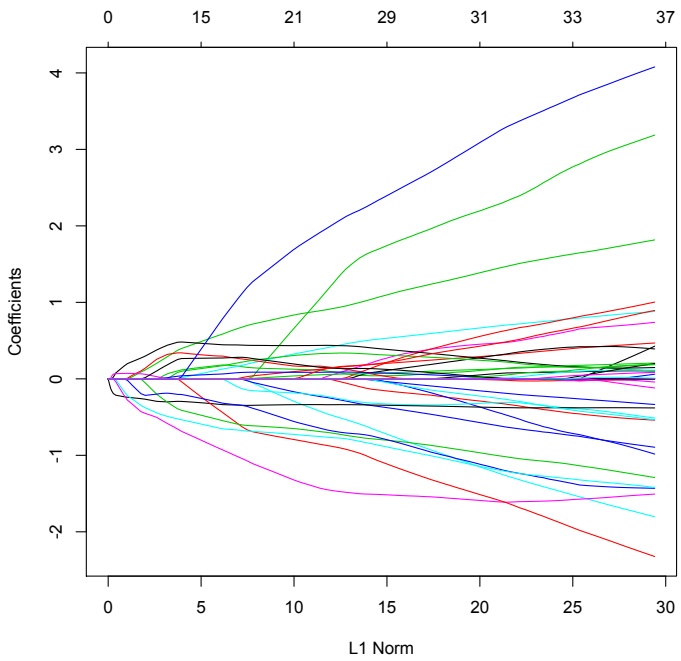

Figure 3: The regularization path illustrates the effect of an increasing parameter λ on the number and the coefficients of the selected features.

For both the IR and the FAIR no working implementations existed in R. The codes can be found in the appendix 9.3 and 9.2. The FAIR method was implemented such that it generalizes the two-class approach conducted by t-tests into a multi-class approach by means of a one-way ANOVA. The optimal number of parameters for the two-class case is analytically derived in Fan and Fan (2008) and an estimator is given. Similar to the approach in NSC and SCRDA, the implementation in 9.2, attempts to find the optimal number of features by means of a 10-fold cross validation.

6 Simulation Experiments

The simulation experiments conducted in this section are supposed to compare the performance of the methods under different stylized conditions. It is obvious that there will not be a method which strictly dominates all other methods in the sense that it always has a superior (prediction) performance. This is clear from the theoretical elaborations of the methods. Every model is based on a set of assumptions and will probably perform excellent when these assumptions are met and will probably fail to different degrees when this is not the case.

A number of issues are supposed to be resolved by means of this simulation. First of all, for any given setting the method yielding the lowest misclassification error is supposed to be identified. Second, the feature selection performance of the methods will be analyzed. Since the signal features are known, it is possible to compare a) the overall number of features selected, b) the number of selected signal features over the overall number of signal features, and c) the number of selected signal features over the number of selected features, indication how many noise features the method selects.

6.1 Settings A - C: Independent Features

Settings A, B, and C simulate a two, three, and five-class classification problem with independent features. It was mentioned earlier that a diagonal covariance matrix is exactly the assumption made in the methods IR, FAIR or NSC. In the realm of microarray data this is a highly unrealistic setting, however, it will serve as a benchmark illustrating how the performance of theses methods will deteriorate when this assumption is relaxed.

For these simulation experiments data matrices $X(n \times p)$ were generated with $p = 2000$ features and $n = 100$ samples. The samples fall into two, three and five classes, with an equal number of observations in every class. Thus, the class-specific sample sizes decrease with an increasing number of groups in settings A, B, and C. In order to study the influence of class-specific sample sizes, the same settings were implemented with the only modification that the class-specific sample sizes took a constant value of 100 features, irrespective of the number of classes. Thus, the sample sizes were 200, 300, and 500 for the two, three, and five-class problem, respectively. These settings were called Ab, Bb, and Cb.

The observations were drawn from a multivariate Gaussian distribution $\boldsymbol{x}_{i\cdot} \sim N(\mu_k, \boldsymbol{I}_p), k = 1, \ldots, K$. Without loss of generality, the first 100 features are signal features defined by $\mu_{jk} \neq \mu_{jl}$ for $j = 1, \ldots, 100; k \neq l$. The means were chosen to be $(-1, 0)$, $(-1, 0, 1)$, and $(-2, -1, 0, 1, 2)$ for the two, three and five-class problems, respectively. The remaining $p - 100$ features are pure noise features satisfying $\mu_{ji} = \cdots = \mu_{jK} = 0$ for $j = 101, \ldots, p$. An illustration of the data matrix can be found in (51). The results of the simulation settings A, B, C, Ab, Bb, and Cb can be found in figures 4 and 7. For settings Ab, Bb, and Cb only the prediction performance was illustrated also in figure 4.

$$
\begin{array}{c}
\overbrace{\phantom{\begin{array}{ccc|ccc} X & \cdots & X & X & \cdots & X \end{array}}}^{\text{signal features}} \quad \overbrace{\phantom{\begin{array}{ccc|ccc} X & \cdots & X & X & \cdots & X \end{array}}}^{\text{noise features}} \\
\left.\begin{array}{l} \mathcal{C}_1 \\ \\ \\ \mathcal{C}_2 \\ \\ \\ \\ \mathcal{C}_2 \\ \\ \\ \end{array}\right\{
\begin{array}{ccc|ccc}
X_{1,1,1} & \cdots & X_{1,100,1} & X_{1,101,1} & \cdots & X_{1,p,1} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
X_{n_2,1,1} & \cdots & X_{n_2,100,1} & X_{n_2,100,1} & \cdots & X_{n_2,p,1} \\
X_{1,1,2} & \cdots & X_{1,100,2} & X_{1,101,2} & \cdots & X_{1,p,2} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
X_{n_1,1,2} & \cdots & X_{n_1,100,2} & X_{n_1,100,2} & \cdots & X_{n_1,p,2} \\
\vdots & & \vdots & \vdots & & \vdots \\
X_{1,1,K} & \cdots & X_{1,100,K} & X_{1,101,K} & \cdots & X_{1,p,K} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
X_{n_K,1,K} & \cdots & X_{n_K,100,K} & X_{n_K,100,K} & \cdots & X_{n_K,p,K}
\end{array}
\left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} \text{n observations} \\
\underbrace{\phantom{\begin{array}{ccc|ccc} X & \cdots & X & X & \cdots & X \end{array}}}_{\text{p features}}
\end{array}
\tag{51}
$$

6.2 Settings D - F: Block-Dependent Features

Settings D, E, and F relax the assumption of independent features and chose a dependence structure which more closely resembles the nature of microarray data. They simulate a two, three, and five-class classification problem with block-dependent features, as suggested in (5) and (6) in section 1.1. The structure of this simulation experiment is similar to the structure in Guo et al. (2005). The objection was to generate a data set which closely resembles real microarray data. $p = 2000$ genes were grouped into $k = 20$ groups each containing $p/k = 100$ genes with an *autoregressive* dependence structure. The closer the two features are within a

block the higher their correlation will be. Whereas the overall sample sizes in settings D, E, and F are 100, three more settings were added that featured constant class-specific sample sizes of 100 observations. These settings are names Db, Eb, and Fb. The results of all block-dependent simulations can be found in figures 5 and 8.

6.3 Settings G - I : Randomly Dependent Features

Settings G, H and I are are two, three and five-class classification problems, respectively, that are based on the assumption that the observations follow a multivariate Gaussian distribution with a random covariance matrix. These settings represent an equally unrealistic assumption about the dependencies of micro-array data as settings A, B, or Cs but lie on the exact opposite of the spectrum of possible structures. The random generation of the covariance matrix was obtained with the `genPositiveDefMat` function in the R package `clusterGeneration`. The results can be found in figures 6 and 9.

6.4 Results

The results of each simulation setting are illustrated in four different plots. The first figure concerns the prediction performance and shows a boxplot of the *rates of correctly classified objects* (RCC). The three remaining figures concern the model selection performance. The second figure shows a boxplot of the cardinality of the set of selected features (FNum). This number alone cannot draw a good picture of the model selection quality since both signal features and noise features can be selected. Therefore the third figure shows the *rate of selected signal features over the total number of signal features* (RSFTsig). Finally, the last figure shows the *rate of selected signal features over the total number of selected features* (RSFTsel).

Not surprisingly, IR, NSC, and SCRDA exhibit an excellent performance in settings A, B, and C, see figure 4. Nearly all observations are correctly classified. SCRDA seems to succeed in finding the right parameter value α that defines the underlying covariance structure. In most of the folds, the correct value of $\alpha = 0$ is obtained. FAIR and RegSVM manage to classify more than 95% of the observations correctly. Surprisingly FAIR seems to improve with an increasing number of classes. The only possible explanation is that FAIR seems to select more features as the number of classes increases. Whereas the performance of NSC and SCRDA is unaffected by the number of classes, the performance of RegSVM and LR.L1 drops dramatically, when the number of classes is augmented. Further inspection into the reasons for this effect revealed that these two methods seemed to have problems with the location of the class centers. While they succeeded in classifying the 'exposed' classes, the 'central' classes could not be identified correctly. Table 1 illustrates a typical example of the contingency tables of RegSVM or LR.L1 comparing the true class-memberships with the predicted ones. This systematic error definitively requires further investigation.

In simulation settings A, B, C, Ab, Bb, and Cb the variance of rcc for IR, NSC, and SCRDA are exceptionally low compared to LR. In all other simulation settings the variances of the classification errors are comparable between methods.

Considering the results of Ab, Bb, and Cb, one must conclude that increasing the class-specific sample sizes results in an improved performance of all methods, however RegSVM and LR.L1 still tend to make the same structural error. The increased sample sizes make the performance differences even more obvious.

Considering the results of settings D, E, and F in figure 5 one has to notice that the performance level deteriorates for all methods and the variance of the rates increases, even for those methods that do not assume independent features. The greatest loss can be observed for

	1	2	3	4	5
1	20	0	0	0	0
2	18	2	0	0	0
3	4	4	7	2	3
4	0	0	0	2	18
5	0	0	0	0	20

Table 1: The rows indicate the true class memberships, whereas the columns indicate the predicted class memberships.

the IR. FAIR, NSC, and SCRDA outperform all other methods loosing only slightly when the number of classes is increased to three or five. This is surprising because the assumption of independent features clearly deviates from the dependence structure in the simulated data set. IR, RegSVM and LR.L1 loose dramatically. The results of Db, Eb, and Fb indicate that an increase in the sample sizes reduce the variance of estimation, the median classification error, however, remains unaltered.

Even more surprisingly, the performances do not further deteriorate when the dependence structure is changed to a completely random covariance matrix, see figure 6. This result clearly indicates that the bias caused by invalid assumption of independent features is more than outweighed by the reduced variance of classification.

From figure 7 one can recognize that FAIR always yields the leanest models with not more than 20 features. Bare in mind that the simulated data contains 100 signal features. It thus only succeeds in selecting no more than one out of five signal features, however not allowing noise features to enter the model. NSC and SCRDA have comparable model sizes of about 50 in the two-class case and one hundred in the three-class case. While SCRDA shows excellent model selection capability in three and five-class case. RegSVM and LR.L1 select more features but do not select more signal features than NSC or SCRDA. The opposite is true, LR.L1 has the worst feature selection performance selecting many more noise features and fewer signal features. The results in 8 confirm these result. When the covariance structure is block-dependent the variance of the model selection accuracy increases, especially for NSC and SCRDA.

7 Real Data Analysis: The COPD Data

A number of pre-processing steps were necessary before tackling the classification task. A robust multi-chip average was calculated, using the `rma` command in the package `affy` to account for differences in the average expression levels between chips. Then a *non-specific* filter was applied to dramatically reduce the number of features. The filter used is *non-specific* in the sense that it does not use phenotype information of the data. It applies a filter based on annotation information (available biological information of the genes) and a variance filter, which filters out those genes that do not show a significant amount of variation across samples and are therefore unlikely to be of any further interest. The `nsFilter` command in the `genefilter` package reduced the number of features from about 50000 to 20069. The following quality control indicated that there were severe batch-effects in the data. The expression values showed a high correlation with the date of the experiment. These batch-effects were removed using a simple rescaling of the data based on a location and spread based model of the data.

A 10-fold cross validation (CV) was conducted with the intention of shedding light on the following two questions:

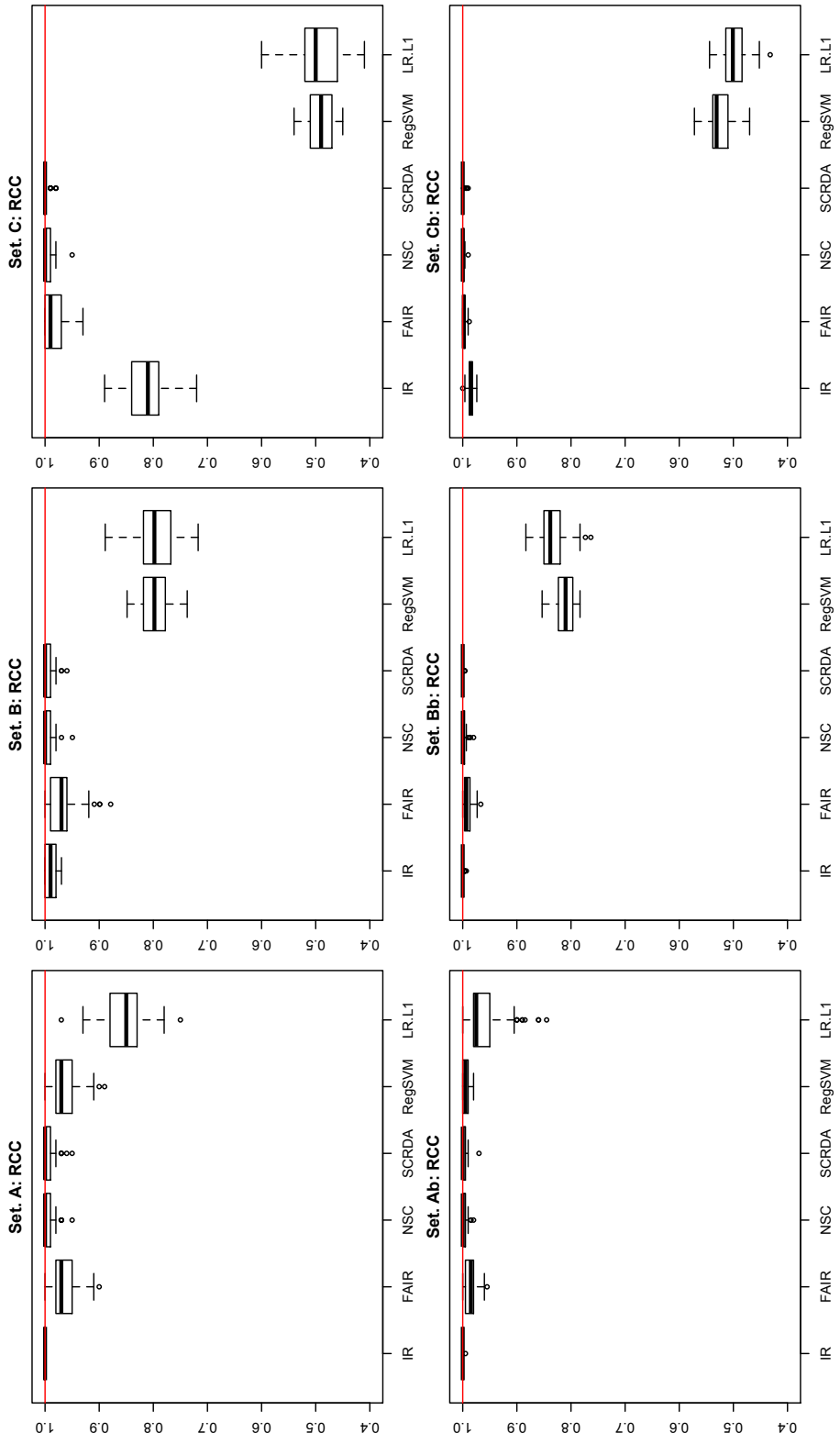

Figure 4: Rates of correctly classified observations of simulation settings A, B, C, Ab, Bb, and Cb.

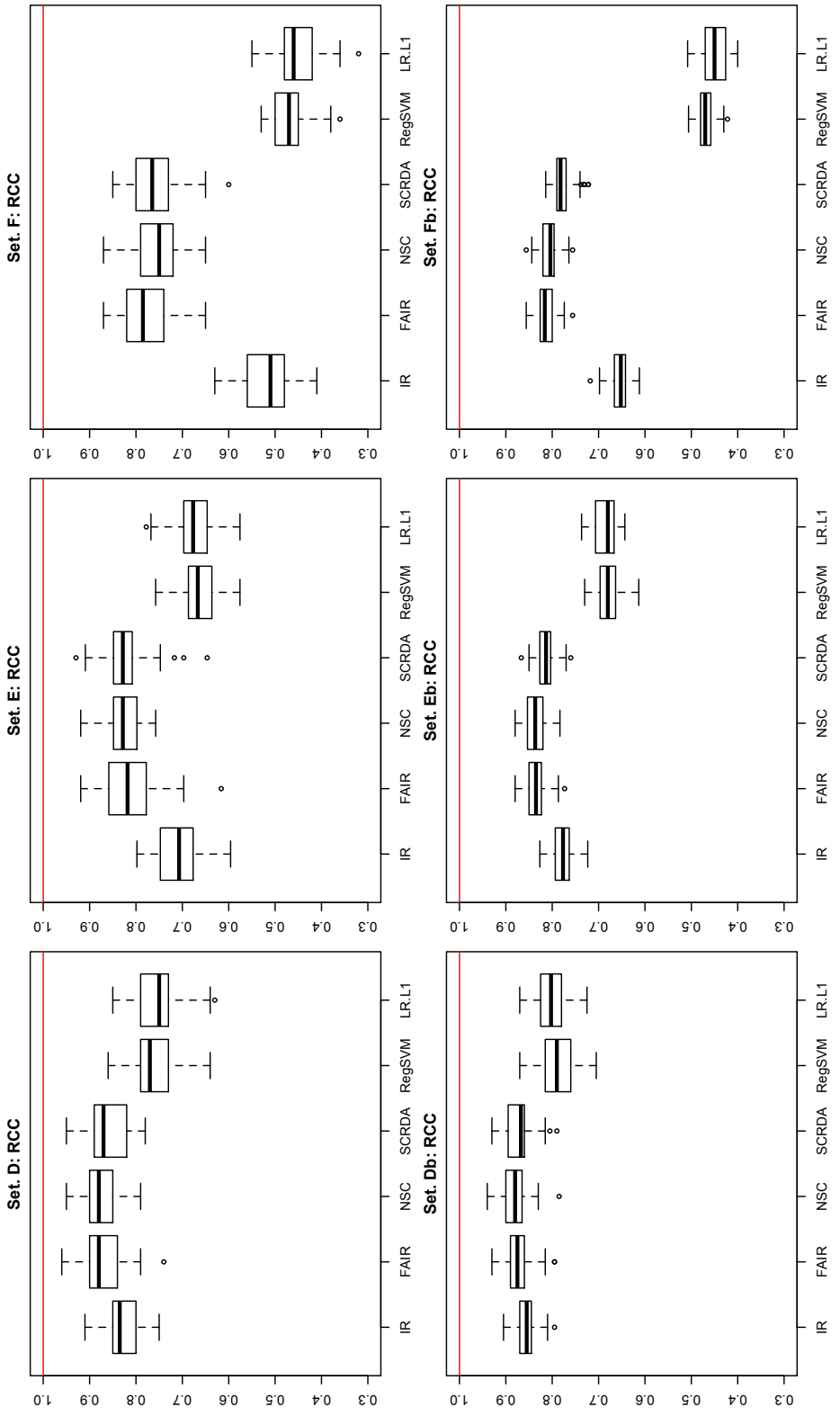

Figure 5: Rates of correctly classified observations of simulation settings D, E, F, Db, Eb, and Fb.

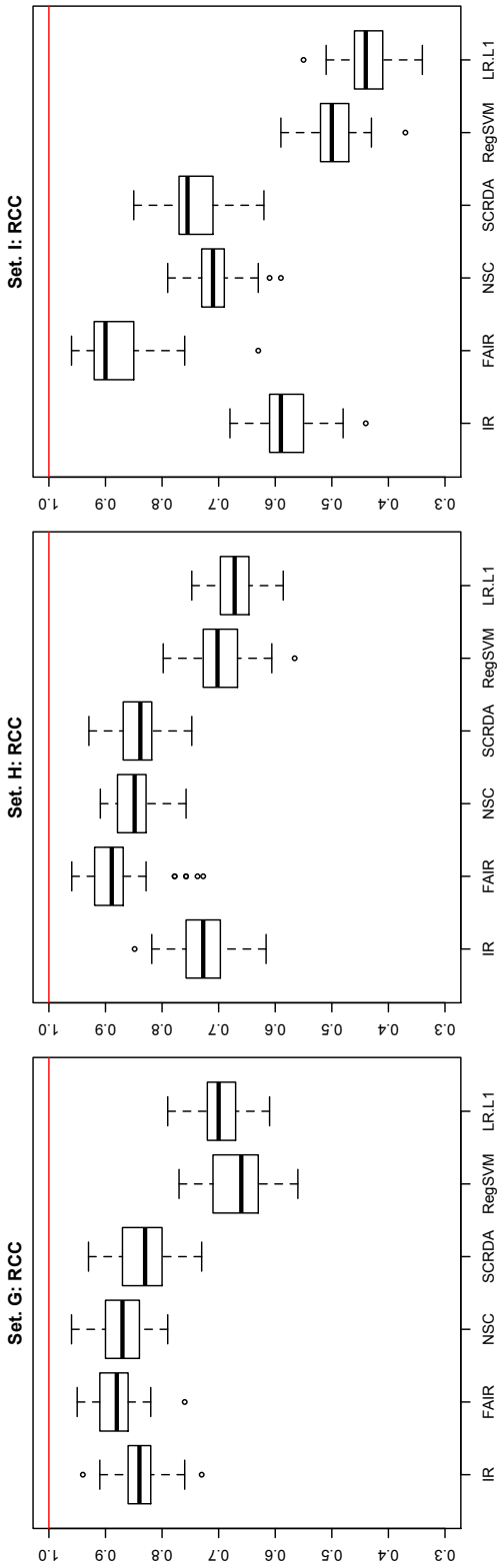

Figure 6: Rates of correctly classified observations of simulation settings G, H, and I.

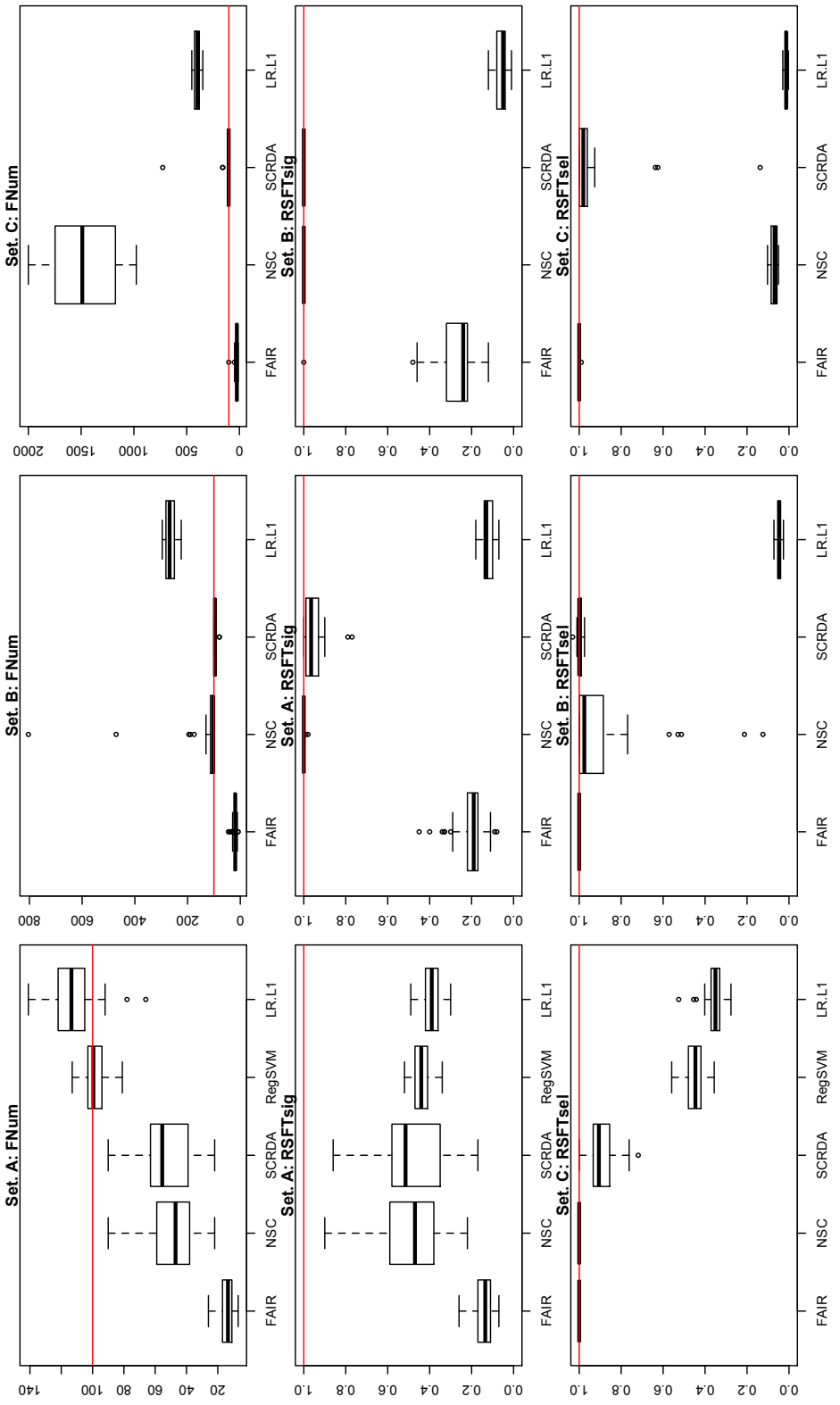

Figure 7: Illustration of the model-selection performances of the designated methods in simulation settings A, B, and C (in columns). The first row depicts the overall number of features selected by each model, the second the number of selected signal features over the total number of signal features and the third depicts the number of signal features over the total number of selected features.

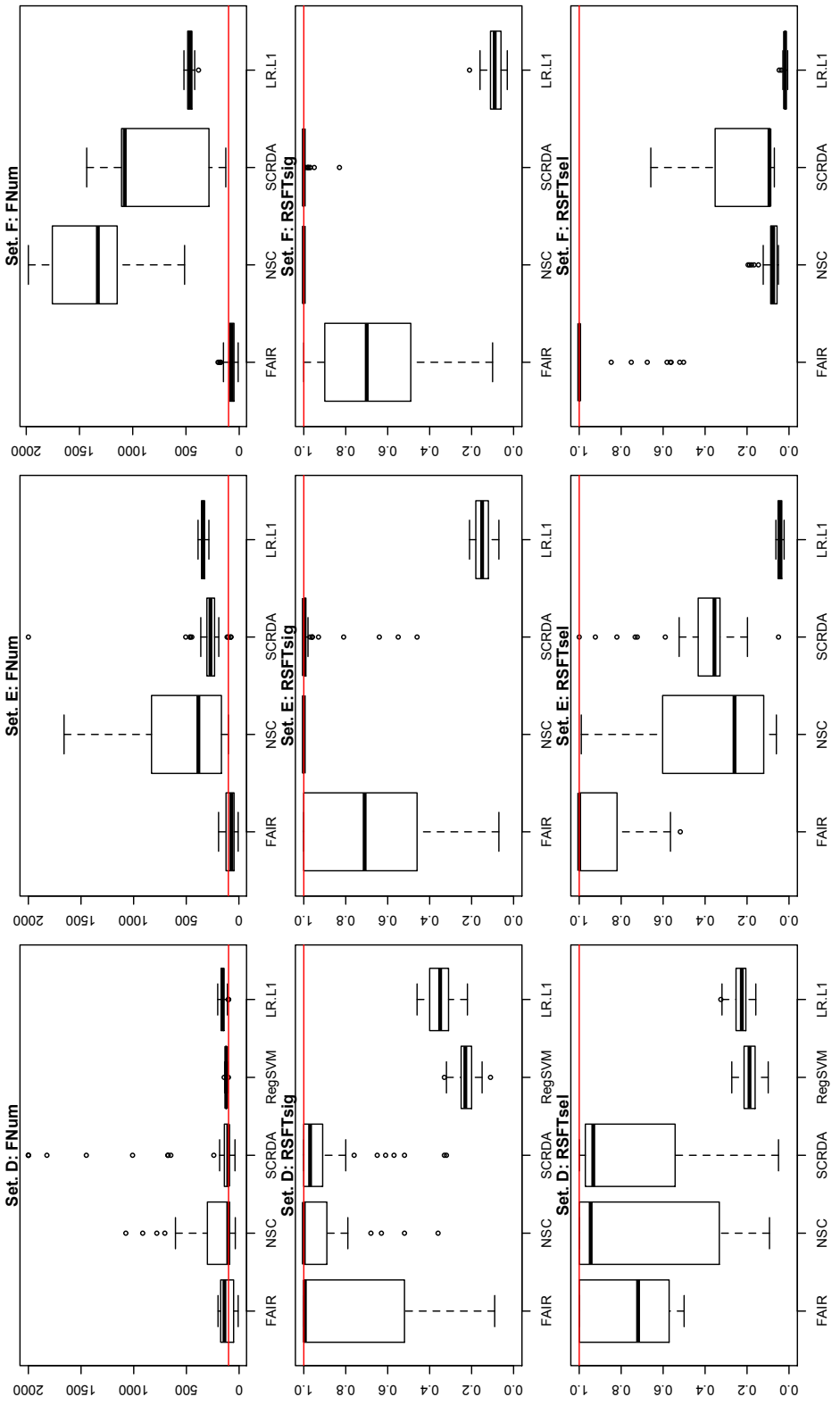

Figure 8: Illustration of the model-selection performances of the designated methods in simulation settings D, E, and F (in columns). The first row depicts the overall number of features selected by each model, the second the number of selected signal features over the total number of signal features and the third depicts the number of signal features over the total number of selected features.

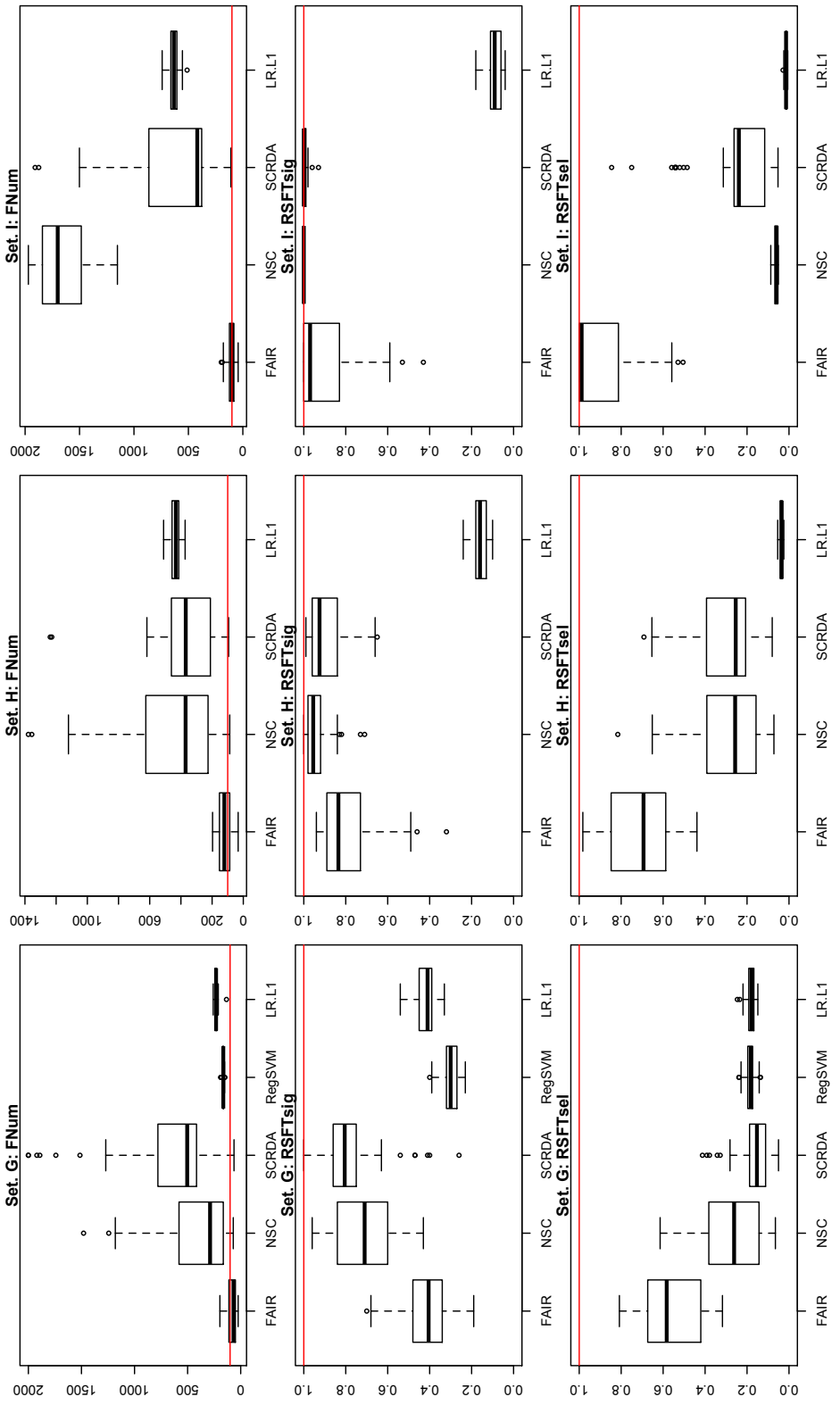

Figure 9: Illustration of the model-selection performances of the designated methods in simulation settings G, H, and I (in columns). The first row depicts the overall number of features selected by each model, the second the number of selected signal features over the total number of signal features and the third depicts the number of signal features over the total number of selected features.

1. Which method has the highest prediction power?

2. How similar are the resulting models, in terms of the set of features that were selected?

The prediction power of the methods is supposed to be estimated by means of the rcc. The second question concerns the interpretability of the resulting models. FAIR, NSC, SCRDA, RegSVM (in the two-class case), and LR.L1 all include different kinds of feature selection. To allow a reasonable interpretation of the model one would demand that the obtained features are more or less independent of the methods used. One would wish for a clear boundary between signal features and noise features. Therefore the feature sets of every fold will be compared.

7.1 The Two-Class Case

The fist classification task attempts to classify between healthy people and patients suffering from COPD. Thus, the sets of samples of all COPD grades were united. This represents the easiest classification task since it features only two classes and one would expect that these classes exhibit the most significant differences in gene-expression values. Eliminating all samples that do not fall into these categories (COPD grade 0) leaves a data set with 82 samples, 18 healthy and 64 patients suffering from COPD. The results of the 10-fold CV are illustrated in Tables 2 and 3

	IR	**FAIR**	**NSC**	**SCRDA**	**RegSVM**	**LR.L1**
Correct classification rate	0.89	0.91	0.85	0.96	0.90	0.94
Median nr. of features	20069	13	10	19335	43	100

Table 2: indicates the misclassification error and the median number of selected features obtained by means of a 10-fold cross validation for the two-class classification problem.

The rates of correctly classified observations differ quite substantially, ranging from 0.85 to 0.96. As one would expect FAIR and SCRDA show a superior performance to IR. NSC, however, yields worse results even, than IR. This is in particularly surprising, since the FAIR and NSC method are very similar. Furthermore, FAIR seems to be competitive to more sophisticated methods such as RegSVM.

Model sizes differ even more dramatically. Whereas NSC or FAIR yield very lean models, containing 10 and 13 features, respectively, and RegSVM and LR.L1 yield reasonable model sizes, SCRDA fails to deliver sparse models. The smallest model obtained during the cross validation contained 16800 features, although SCRDA allows to shrink the centroids. At the same time SCRDA showed the best performance with a classification rate of 0.96. How can this be interpreted? An internal cross validation determines two parameters α and δ. The first accounts for the numerical instability caused by inverting a singular covariance matrix, and the other (shrinkage parameter) accounts for the effect of noise accumulation. By looking at the optimal parameter sets one will detect that the values of the shrinkage parameter δ are very small and that α is not stable. Its values alternates between 0.11 and 0.99, two completely different covariance structures. Thus, the conclusion has to be drawn that the numerical effect is stronger and that most features contain at least a small effect. The validity of this conclusion is open for further research.

The question how similar the resulting models are is supposed to be answered in table 3. The diagonal elements indicate the number of the features that were selected in more than five

of the 10 CV folds. The off-diagonal elements indicate the cardinality of the intersecting sets of the two methods.

In general the models consist of completely different features. The cardinality of the intersection sets is negligible compared to the cardinality of the feature sets. An interesting observation is that the set of features selected by RegSVM is a subset of the set of features selected by LR.L1 which yields models with twice as many features on average.

	FAIR	**NSC**	**SCRDA**	**RegSVM**	**LR.L1**
FAIR	12	4	12	2	3
NSC	4	10	10	4	6
SCRDA	12	10	19996	34	69
RegSVM	2	4	34	34	32
LR.L1	3	6	69	32	69

Table 3: indicates the cardinality of intersecting sets of selected features gained by different classification methods for the 2-class classification problem.

7.2 The Three-Class Case

An extension of the classification task is to include the 'at risk' patients, which by current medical means of diagnosis cannot be clearly classified as healthy or suffering from COPD. The results can be found in tables 4 and 5.

	IR	**FAIR**	**NSC**	**SCRDA**	**RegSVM**	**LR.L1**
Correct classification rate	0.53	0.64	0.53	0.92	0.82	0.86
Median Nr. of Features	20069	10	25	19946	20069	538

Table 4: indicates the misclassification error and the median number of selected features obtained by means of a 10-fold cross validation for the 3-class classification problem.

	FAIR	**NSC**	**SCRDA**	**LR.L1**
FAIR	9	3	9	0
NSC	3	24	24	0
SCRDA	9	24	20066	112
LR.L1	0	0	112	112

Table 5: indicates the cardinality of intersecting sets of selected features gained by different classification methods for the 3-class classification problem.

As expected all methods lose predictive power, but while for some methods this loss is reasonable, IR, FAIR, and NSC loose dramatically. Bear in mind that for a three-class classification problem random guessing results in a correct-classification rate of 1/3. SCRDA remains the leader in terms of performance, with an excellent rate of 0.92. RegSVM and LR.L1 only lose 0.08.

FAIR and NSC still yield very lean models, but considering their predictive performance one must conclude that they are too restrictive. The model size of LR.L1 increased about five-fold

as compared to the two-class case. RegSVM selects the full feature set since L1 regularization is not available for multi-class classification problems. Therefore, only four methods are included in table 5, which tells the following story. Whereas, the model sets for FAIR and NSC are relatively stable. The median number of features are 27 and 23 for FAIR and NSC, 20 of these features are included in more than 5 of the 10 CV folds. This is definitively not the case for SCRDA and LR.L1.

7.3 The Five-Class Case

The most demanding classification task is to differentiate between different grades of COPD. These are classified using the GOLD index ranging from 0 to 4. Since the sample included too few observations of class COPD IV, they were added to class COPD III. Together with the class of healthy people this makes five classes. The classification results can be found in tables 6 and 7.

	IR	**FAIR**	**NSC**	**SCRDA**	**RegSVM**	**LR.L1**
Correct classification rate	0.42	0.52	0.14	0.18	0.76	0.86
Median Nr. of Features	20069	27	23	129	20069	573

Table 6: indicates the misclassification error and the median number of selected features obtained by means of a 10-fold cross validation for the 5-class classification problem.

	FAIR	**NSC**	**SCRDA**	**LR.L1**
FAIR	20	4	0	0
NSC	4	20	4	0
SCRDA	0	4	49	0
LR.L1	0	0	0	72

Table 7: indicates the cardinality of intersecting sets of selected features gained by different classification methods for the five-class classification problem.

LR.L1 turned out to be the winner of this classification task. Surprisingly, this method did not loose any prediction accuracy as compared to the three-class case. Also RegSVM delivers good results. The most surprising fact is that the performance of SCRDA crashed, from being the top performer to performing worse than random guessing, which in this setting is equivalent to a correctly classifying one out of five. For the first time the method delivered a sparse model. For the moment I have no explanation for this behavior. NSC performs even worse, far below random guessing. Except for SCRDA model sizes increased slightly but the relative sizes were left unchanged.

8 Conclusion

High dimensional feature spaces have become the typical characteristic of data in many modern fields of science such as genomics. We have seen how different classical statistical methods have been modified to account for this characteristic. The modification have in common that they can be interpreted as introducing a bias with the intention of reducing the prediction error. The modifications include the assumption of independent features, feature selection based on one-way ANOVA, or the introduction of regularization terms. In some situations the resulting bias is more than offset by the reduced variance, thus, improving prediction power. The feasibility of these modifications are, of course dependent of the data structure. Simulation experiments were conducted to test these methods in a myriad of different settings, the parameters of which included the signal-to-noise ratio in every feature, the number of signal features, the dependence structure of the data, the number of class-specific and overall observations. These parameters where varied as to gain insight into the strengths and weakness of the proposed methods in order to support the decision which method should finally be applied. The simulation results have, as expected, shown, that there is not one method which dominates all others, in the sense that its prediction performance is always superior, but rather that in different situations different methods outperform all others. It is one of the most challenging tasks of the applied researcher to decide which method to use. This decision can of course only be based on experience, simulation results and an inspection of the data at hand.

The simulation results confirmed the theoretical results of section 2.6 in the sense that methods that make the incorrect assumption of independent features can in certain situations outperform methods, which do not make this assumption. However, it was surprising to see that this can still be the case when there were strong linear dependencies and the dependence structure was completely random.

It was further surprising to see that SCRDA did not yield sparse models but at the same time delivered the best prediction results. It remains an open question, whether it can be concluded that the dependence structure is not sparse.

Also the excellent performance of the LR.L1 method in the five-class real data cross validation remains an open question. This kind of result could not be obtained with simulated data.

9 Appendix

9.1 R Code: Simulation

```
generateData <- function(n.train=c(50,50), n.test=c(50,50), cor='block-dependent', means=c
    (-2,2), rho=c(0.9,0.9), signal.feat=20, block.size=c(10,10), block.number=c(10,10), feat
    =1000) {

       # is es sinnvoll unterschiedliche kovarianzen zwischen den klasssen zuzulassen?

       stopifnot(length(n.train)==length(n.test)&length(n.test)==length(means))
       stopifnot(sum(block.size*block.number)==(length(block.size)*block.size[1]*block.number
           [1]))


       k <- length(means)
       if (cor=='independent') {p <- feat}
       else {p <- block.size[1]*block.number[1]}
       Sigma <- vector('list',k)
       mu <- vector('list',k)
       y.train <- vector(mode='numeric')
       X.train <- vector(mode='numeric')
       y.test <- vector(mode='numeric')
       X.test <- vector(mode='numeric')



       if (cor =='independent') {

               for (i in 1:k) {
               Sigma[[i]] <- diag(rep(1,p))
               mu[[i]] <- c(rep(means[i],signal.feat), rep(0,p-signal.feat))
                                   }

       }

       if (cor == 'block-dependent') {

        SigGenes <- function(rho, B, m) {
       SS <- array(0, dim = c(B, B, m))
       for (h in 1:m) {
           if (h%%2 == 1)
               r = rho
           else r = -rho
           SS[, , h] <- toeplitz(r^(0:(B - 1)))
       }
       S <- diag(0, m * B)
       for (h in 1:m) for (i in 1:B) for (j in 1:B) S[(h - 1) *
           B + i, (h - 1) * B + j] <- SS[i, j, h]
       S
   }

       # generate parameters of a multivariate Gaussian distribution

       for (i in 1:k) {
               Sigma[[i]] <- SigGenes(rho[i], block.size[i], block.number[i])
               mu[[i]] <- c(rep(means[i],signal.feat), rep(0,p-signal.feat))
                                   }
```

```
        }

        # generate data--matrix

        for (i in 1:k) {
                X.train <- rbind(X.train, mvrnorm(n.train[i], mu[[i]], Sigma[[i]]))
                y.train <- c(y.train, rep(i,n.train[i]))
                X.test <- rbind(X.test, mvrnorm(n.test[i], mu[[i]], Sigma[[i]]))
                y.test <- c(y.test, rep(i,n.test[i]))
                                        }

                colnames(X.train) <- c(1:dim(X.train)[2])

        return(list('p'=p, 'n.train'=n.train, 'n.test'=n.test, 'signal.features'=signal.feat, '
            means'=means, 'cor'=cor, 'y.train'=y.train, 'X.train'=X.train, 'y.test'=y.test, 'X.
            test'=X.test))



}
```

R-Code 1: Data Simulation

```
SIM <- function (methods=c('IR', 'FAIR', 'NSC', 'SCRDA', 'SVM', 'RegSVM', 'LR.L1', 'LR.ISIS'),
    reps=50, cor='block-dependent', n.train=c(50,50), n.test=c(50,50), means=c(-1,0), signal.
    feat=100, block.size=c(10,10), block.number=c(10,10), rho=c(0.9,0.9)) {

        obj <- vector('list', length=4+reps)
        nms <- vector('character', reps)
        for (i in 1:reps) {nms[i] <- paste('set',i, sep='')}
        names(obj) <- c('methods', 'cor', 'classes', 'reps', nms)

        for (i in 1:reps) {

                print(i)
                set <- generateData(n.train=n.train, n.test=n.test, cor=cor, means=means, feat
                    =2000, signal.feat=signal.feat, block.size=block.size, block.number=block.
                    number, rho=rho)
                print('generated')
                res.set <- vector('list', length=length(methods))
                names(res.set) <- methods

                if ('IR' %in% methods) {
                        tmp <- DA.IR(y.train=set$y.train , X.train=set$X.train, X.test=set$X.
                            test)
                        res.set[['IR']] <- c(tmp, validate(tmp$y.prediction, set$y.test))
                                                }

        if ('FAIR' %in% methods) {
                        tmp <- DA.FAIR(y.train=set$y.train , X.train=set$X.train, X.test=set$X.
                            test)
                res.set[['FAIR']] <- c(tmp, validate(tmp$y.prediction, set$y.test))
                                                }

                if ('NSC' %in% methods) {
                        tmp <- DA.NSC(y.train=set$y.train , X.train=set$X.train, X.test=set$X.
                            test)
```

```
                res.set[['NSC']] <- c(tmp, validate(tmp$y.prediction, set$y.test))
                                        }

                if ('SCRDA' %in% methods) {
                        tmp <- DA.SCRDA(y.train=set$y.train , X.train=set$X.train, X.test=set$X
                            .test)
                res.set[['SCRDA']] <- c(tmp, validate(tmp$y.prediction, set$y.test))
                                        }

                if ('RegSVM' %in% methods) {
                        tmp <- RegSVM(y.train=set$y.train , X.train=set$X.train, X.test=set$X.
                            test)
                res.set[['RegSVM']] <- c(tmp, validate(tmp$y.prediction, set$y.test))
                                        }

                if ('LR.L1' %in% methods) {
                        tmp <- LR.L1(y.train=set$y.train , X.train=set$X.train, X.test=set$X.
                            test)
                res.set[['LR.L1']] <- c(tmp, validate(tmp$y.prediction, set$y.test))
                                        }


                obj[[4+i]] <- res.set

                                        }

        obj[['methods']] <- methods
        obj[['cor']] <- cor
        obj[['classes']] <- length(means)
        obj[['reps']] <- reps

        return(obj)

        }
```

R-Code 2: IR

```
# SIM01: Independent, 2 Class

res.sim01 <- SIM(cor='independent', reps=50, n.train=c(50,50), n.test=c(50,50), means=c(-1,0),
    signal.feat=100)

res.sim01b <- SIM(cor='independent', reps=50, n.train=c(100,100), n.test=c(100,100), means=c
    (-1,0), signal.feat=100)

# SIM02: Independent, 3 Class

res.sim02 <- SIM(cor='independent', reps=50, n.train=c(33,33,33), n.test=c(33,33,33), means=c
    (-1,0,1), signal.feat=100)

res.sim02b <- SIM(cor='independent', reps=50, n.train=c(100,100,100), n.test=c(100,100,100),
    means=c(-1,0,1), signal.feat=100)

# SIM03: Independent, 5 Class

res.sim03 <- SIM(cor='independent', reps=50, n.train=c(20,20,20,20,20), n.test=c
```

```
    (20,20,20,20,20), means=c(-2,-1,0,1,2), signal.feat=100)

res.sim03b <- SIM(cor='independent', reps=50, n.train=c(100,100,100,100,100), n.test=c
    (100,100,100,100,100), means=c(-2,-1,0,1,2), signal.feat=100)

# SIM04: Block–dependent, 2 Class

res.sim04 <- SIM(cor='block-dependent', reps=50, n.train=c(50,50), n.test=c(50,50), means=c
    (-1,0), rho=c(0.9,0.9), block.size=c(100,100), block.number=c(20,20), signal.feat=100)

res.sim04b <- SIM(cor='block-dependent', reps=50, n.train=c(100,100), n.test=c(100,100), means=
    c(-1,0), rho=c(0.9,0.9), block.size=c(100,100), block.number=c(20,20), signal.feat=100)

# SIM05: Block–dependent, 3 Class

res.sim05 <- SIM(cor='block-dependent', reps=50, n.train=c(33,33,33), n.test=c(33,33,33), means
    =c(-1,0,1), block.size=c(100,100,100), block.number=c(20,20,20), rho=c(0.9,0.9,0.9), signal
    .feat=100)

res.sim05b <- SIM(cor='block-dependent', reps=50, n.train=c(100,100,100), n.test=c(100,100,100)
    , means=c(-1,0,1), block.size=c(100,100,100), block.number=c(20,20,20), rho=c(0.9,0.9,0.9),
     signal.feat=100)

# SIM06: Block–dependent, 5 Class

res.sim06 <- SIM(cor='block-dependent', n.train=c(20,20,20,20,20), n.test=c(20,20,20,20,20),
    means=c(-2,-1,0,1,2), rho=c(0.9,0.9,0.9,0.9,0.9), block.size=c(100,100,100,100,100), block.
    number=c(20,20,20,20,20), signal.feat=100)

res.sim06b <- SIM(cor='block-dependent', n.train=c(100,100,100,100,100), n.test=c
    (100,100,100,100,100), means=c(-2,-1,0,1,2), rho=c(0.9,0.9,0.9,0.9,0.9), block.size=c
    (100,100,100,100,100), block.number=c(20,20,20,20,20), signal.feat=100)

# SIM07: Random, 2–Class

res.sim07 <- SIM(cor='random', reps=50, feat=2000, means=c(-1,0), n.train=c(50,50), n.test=c
    (50,50))

res.sim07b <- SIM(cor='random', reps=50, feat=2000, means=c(-1,0), n.train=c(100,100), n.test=c
    (100,100))

# SIM08: Random, 3–Class

res.sim08 <- SIM(cor='random', reps=50, feat=2000, means=c(-1,0,1), n.train=c(33,33,33), n.test
    =c(33,33,33))

res.sim08b <- SIM(cor='random', reps=50, feat=2000, means=c(-1,0,1), n.train=c(100,100,100), n.
    test=c(100,100,100))

# SIM09: Random, 5– Class

res.sim09 <- SIM(cor='random', reps=50, feat=2000, means=means=c(-2,-1,0,1,2), n.train=c
    (20,20,20,20,20), n.test=c(20,20,20,20,20))

res.sim09b <- SIM(cor='random', reps=50, feat=2000, means=means=c(-2,-1,0,1,2), n.train=c
    (100,100,100,100,100), n.test=c(100,100,100,100,100))
```

R-Code 3: IR

9.2 R Code: FAIR

```
DA.FAIR <- function(y.train, X.train, X.test, maxm=200) {

        n <- length(y.train)
        k <- y.train[which.max(y.train)]

        # Apply gene-wise Kruskal Tests

        #kt <- function(x) {return(kruskal.test(x, g=y.train)$p.value)}
        #kruskal.tests <- apply(X.train,2,kt)
        #seq <- order(kruskal.tests, decreasing=F)

        # Apply gene-wise one-way ANOVA

        anovas <- function(x) {anova(lm(x~y.train))$F[1]}
        seq <- order(apply(X.train,2,anovas), decreasing=T)

        # Find optimal number of features m by cross validation

        MC <- rep(0,min(maxm,dim(X.train)[2]))
    o <- sample(1:n)
    leave.out <- trunc(n/10)
        groups <- vector("list", 10)
        for (j in 1:9) {
            jj <- (1 + (j - 1) * leave.out)
            groups[[j]] <- (o[jj:(jj + leave.out - 1)])
        }
        groups[[10]] <- o[(1 + (10 - 1) * leave.out):n]



        for (l in 1:10) {

                mu <- vector('list',k)
                pi <- rep(0,k)

                train <- !(c(1:n)%in%groups[[l]])
                test <- c(1:n)%in%groups[[l]]

                X.sub <- X.train[train,]
                y.sub <- y.train[train]
                X.sub.test <- X.train[test,]
                y.sub.test <- y.train[test]

                for (i in 1:k) {mu[[i]] <- apply(X.sub[(y.sub==i),],2,mean)
                                pi[i] <- sum((y.sub==i))/length(y.sub)
                        }
                var.pooled <- apply(X.sub,2,var)

                for (m in 1:min(maxm,dim(X.train)[2])) {
                delta <- matrix(rep(0,length(y.sub.test)*k), ncol=k)

                for (i in 1:length(y.sub.test)) {
                        for (j in 1:k) {

                                delta[i,j] <- - sum((X.sub.test[i,seq[1:m]] - mu[[j]][seq[1:m
                                    ]])^2/var.pooled[seq[1:m]]) + 2*log(pi[j])
```

```
                                                }
                                        }
                        class <- apply(delta,1,which.max)
                        #print(sum(class!=y.sub.test))
                        MC[m] <- MC[m] + sum(class!=y.sub.test)

                                                }
                                }

        # Select best model size

        tmp <- MC[which.min(MC):length(MC)]==min(MC)
        prodton <- function(x,n) {prod(x[1:n])}
        tmp2 <- rep(0, length(tmp))
        for (i in 1:length(tmp)) {tmp2[i] <- prodton(tmp,i)}
        m <- trunc(which.min(MC)+(which.min(tmp2)-1)/2)

        # Estimate parameters

        mu <- vector('list',k)
        pi <- rep(0,k)
        for (i in 1:k) {mu[[i]] <- apply(X.train[(y.train==i),],2,mean)
                                        pi[i] <- sum((y.train==i))/length(y.train)
                        }
        var.pooled <- apply(X.train,2,var)

        opt <- seq[1:m]

        # Predict

        if (is.vector(X.test)) {n <- 1}
        else {n <- dim(X.test)[1]}


        delta <- matrix(rep(0,n*k), nrow=n)
        for (i in 1:n){
                for (j in 1:k) {

                        if (is.vector(X.test)) {delta[i,j] <- - sum((X.test[opt] - mu[[j]][opt
                            ])^2/var.pooled[opt]) + 2*log(pi[j])}
                        else {delta[i,j] <- - sum((X.test[i,opt] - mu[[j]][opt])^2/var.pooled[
                            opt]) + 2*log(pi[j])}

                                                }

                                        }

        class <- apply(delta,1,which.max)
        return(list('y.prediction'=class, 'features.number'=m, 'features.list'=colnames(X.train
            )[opt]))

}
```

R-Code 4: FAIR

9.3 R Code: IR

```
DA.IR <- function(y.train, X.train, X.test) {

        k <- y.train[which.max(y.train)]
        mu <- vector('list',k)
        pi <- rep(0,k)
        for (i in 1:k) {mu[[i]] <- apply(X.train[(y.train==i),],2,mean)
                                    pi[i] <- sum((y.train==i))/length(y.train)
                        }
        var.pooled <- apply(X.train,2,var)



        if (is.vector(X.test)) {n <- 1}
        else {n <- dim(X.test)[1]}

        delta <- matrix(rep(0,n*k), nrow=n)
        for (i in 1:n){
                for (j in 1:k) {

                        if (is.vector(X.test)) {delta[i,j] <- - sum((X.test[i] - mu[[j]])^2/var
                            .pooled) + 2*log(pi[j])}
                        else{delta[i,j] <- - sum((X.test[i,] - mu[[j]])^2/var.pooled) + 2*log(
                            pi[j]) }

                                                }

                                    }

        class <- apply(delta,1,which.max)

        return(list('y.prediction'=class, 'features.number'=dim(X.train)[2], 'features.list'=
            colnames(X.train)))

}
```

R-Code 5: IR

9.4 R Code: Wrapper Functions

```
DA.NSC <- function(y.train, X.train, X.test) {

        data <- list(x=t(X.train), y=y.train, genenames=colnames(X.train))
        res.train <- pamr.train(data=data)
        res.cv <- pamr.cv(res.train, data=data)
        opt.thres <- res.cv$threshold[max(which(res.cv$error==min(res.cv$error)))]
        features <- which(colnames(X.train)%in%rownames(gene.list(fit=res.train, data=data,
            threshold=opt.thres, genenames=T)))
        y.prediction <- pamr.predict(fit=res.train, newx=t(X.test), threshold=opt.thres)

        return(list('y.prediction'=y.prediction, 'features.number'=res.cv$size[max(which(res.cv
            $error==min(res.cv$error)))], 'features.list'=colnames(X.train)[features]))

}
```

R-Code 6: IR

```
DA.SCRDA <- function(y.train, X.train, X.test) {

                        res.RDA <- rda(x=t(X.train), y=y.train, genelist=T)
                res.RDA.CV <- rda.cv(res.RDA, x=t(X.train), y=y.train)
                opt <- which(res.RDA.CV$cv.err==min(res.RDA.CV$cv.err), arr.ind=T)
                minmin <- opt[which.min(res.RDA.CV$ngene[opt]),]
                alpha.opt <- res.RDA.CV$alpha[minmin[1]]
                delta.opt <- res.RDA.CV$delta[minmin[2]]
                predict.RDA <- predict.rda(res.RDA, x=t(X.train), y=y.train, xnew=t(X.test),
                    alpha=alpha.opt, delta=delta.opt, genelist=T)
                return(list('y.prediction'=predict.RDA, 'alpha.opt'=alpha.opt, 'delta.opt'=
                    delta.opt, 'features.number'= res.RDA.CV$ngene[minmin[1], minmin[2]], '
                    features.list'=colnames(X.train)[which(res.RDA$gene.list[minmin[1], minmin
                    [2],]==1)]))
}
```

R-Code 7: IR

```
RegSVM <- function(y.train, X.train, X.test) {

                        if (length(unique(y.train))>2) {type=4}
                        else {type=5}

                        res <- LiblineaR(data=X.train, labels=y.train, type=type)
                        prediction <- predict(object=res, newx=X.test)$predictions
                feat.list <- colnames(X.train)[which(res$W!=0)]

                return(list('y.prediction'=prediction, 'features.list'=feat.list, 'features.
                    number'=length(feat.list)-1))
}
```

R-Code 8: IR

```
LR.L1 <- function(y.train, X.train, X.test) {

                        res <- LiblineaR(data=X.train, labels=y.train, type=6)
                        prediction <- predict(object=res, newx=X.test)$predictions
                        feat.list <- colnames(X.train)[which(res$W!=0)]

                tmp <- vector(mode='character')
                for (i in 1:dim(res$W)[1]) {
                        tmp <- c(tmp, colnames(X.train)[which(res$W[i,]!=0)])
                }

                return(list('y.prediction'=prediction, 'features.list'=feat.list, 'features.
                    number'=length(unique(tmp))))
}
```

R-Code 9: IR

9.5 R Code: Cross Validation

```
CV <- function(y,X,folds=10, methods=c('IR', 'FAIR', 'NSC', 'SCRDA', 'SVM', 'RegSVM', 'LR.L1',
    'LR.ISIS'),file='CV.RData') {

        n <- length(y)
        call <- match.call()

        if (folds == n) {groups <- c(1:n)}

    if (folds < n) {
        leave.out <- trunc(n/folds)
        o <- sample(1:n)

        groups <- vector("list", folds)
        for (j in 1:(folds - 1)) {
            jj <- (1 + (j - 1) * leave.out)
            groups[[j]] <- (o[jj:(jj + leave.out - 1)])
        }
        groups[[folds]] <- o[(1 + (folds - 1) * leave.out):n]
                                        }

        CV <- vector('list', length(methods)+8)
        names(CV) <- c('y','X','n.obs','n.feat','groups','folds','call','methods', methods)

    names <- vector('character', folds)
        for (i in 1:folds) {names[i] <- paste('fold',i, sep='')}


    if ('IR' %in% methods) {
        print('IR')
        res.IR <- vector('list', folds)
        for (j in 1:folds) {
            print(j)
                res.IR[[j]] <- DA.IR(y.train=y[-groups[[j]]] , X.train=X[-groups[[j]],], X.test
                    =X[groups[[j]],])
                                                }
        names(res.IR) <- names
        CV[['IR']] <- res.IR
                                                }


    if ('FAIR' %in% methods) {
        print('FAIR')
        res.FAIR <- vector('list', folds)
        for (j in 1:folds) {
                print(j)
                res.FAIR[[j]] <- DA.FAIR(y.train=y[-groups[[j]]] , X.train=X[-groups[[j]],], X.
                    test=X[groups[[j]],])
                                                }
        names(res.FAIR) <- names
        CV[['FAIR']] <- res.FAIR
                                                }


    if ('NSC' %in% methods) {
        print('NSC')
        res.NSC <- vector('list', folds)
        for (j in 1:folds) {
                print(j)
```

```
            res.NSC[[j]] <- DA.NSC(y.train=y[-groups[[j]]] , X.train=X[-groups[[j]],], X.
                test=X[groups[[j]],])
                                        }
    names(res.NSC) <- names
    CV[['NSC']] <- res.NSC
                                        }


if ('SCRDA' %in% methods) {
    print('SCRDA')
    res.SCRDA <- vector('list', folds)
    for (j in 1:folds) {
            print(j)
            res.SCRDA[[j]] <- DA.SCRDA(y.train=y[-groups[[j]]], X.train=X[-groups[[j]],], X
                .test=X[groups[[j]],])
                                        }
    names(res.SCRDA) <- names
    CV[['SCRDA']] <- res.SCRDA
                                                }


 if ('SVM' %in% methods) {
    print('SVM')
    res.SVM <- vector('list', folds)
    for (j in 1:folds) {
            print(j)
            res.SVM[[j]] <- SVM(y.train=y[-groups[[j]]], X.train=X[-groups[[j]],], X.test=X
                [groups[[j]],])
                                        }
    names(res.SVM) <- names
    CV[['SVM']] <- res.SVM
                                                }


    if ('RegSVM' %in% methods) {
    print('RegSVM')
    res.RegSVM <- vector('list', folds)
    for (j in 1:folds) {
            print(j)
            res.RegSVM[[j]] <- RegSVM(y.train=y[-groups[[j]]], X.train=X[-groups[[j]],], X.
                test=X[groups[[j]],])
                                        }
    names(res.RegSVM) <- names
    CV[['RegSVM']] <- res.RegSVM
                                                }


 if ('LR.L1' %in% methods) {
    print('LR.L1')
    res.LR.L1 <- vector('list', folds)
    for (j in 1:folds) {
            print(j)
            res.LR.L1[[j]] <- LR.L1(y.train=y[-groups[[j]]], X.train=X[-groups[[j]],], X.
                test=X[groups[[j]],])
                                        }
    names(res.LR.L1) <- names
    CV[['LR.L1']] <- res.LR.L1
                                                }
```

```
    if ('LR.ISIS' %in% methods) {
        print('LR.ISIS')
        res.LR.ISIS <- vector('list', folds)
        for (j in 1:folds) {
                print(j)
                res.LR.ISIS[[j]] <- LR.ISIS(y.train=y[-groups[[j]]], X.train=X[-groups[[j]],],
                    X.test=X[groups[[j]],])
                                                }
        names(res.LR.ISIS) <- names
        CV[['LR.ISIS']] <- res.LR.ISIS
                                                        }



    CV[['y']] <- y
    CV[['X']] <- X
    CV[['n.obs']] <- length(y)
    CV[['n.feat']] <- dim(X)[2]
    CV[['groups']] <- groups
    CV[['folds']] <- folds
    CV[['call']] <- call
    CV[['methods']] <- methods


  save(CV, file=file)
  return(CV)

}
```

R-Code 10: IR

```
evalCV <- function(obj) {

        # Misclassification rate

        MCrate <- rep(0,length(obj$methods))
        seq <- rep(0,0)
        for (i in 1: obj$folds) {seq <- c(seq,obj$groups[[i]])}

        # Rating of features

        rating <- matrix(rep(0,length(obj$methods)*obj$n.feat), nrow=length(obj$methods))
        rownames(rating) <- obj$methods

        # Average number and variance of features

        feat <- matrix(rep(0,obj$folds*length(obj$methods)), ncol=length(obj$methods))
        colnames(feat) <- obj$methods

        for (i in 1:length(obj$methods)) {
                pred <- rep(0,0)

        for (j in 1: obj$folds) {

                        pred <- c(pred,obj[[obj$methods[i]]][[j]]$y.prediction)
                        rating[i,] <- rating[i,] + (colnames(obj$X)%in%obj[[obj$methods[i]]][[j
                            ]]$features.list)*1
                        feat[j,i] <- obj[[obj$methods[i]]][[j]]$features.number
```

```
                        }

                MCrate[i] <- sum(obj$y[seq]==pred)/length(obj$y)
        }

        AVF <- apply(feat,2,mean)
        SDF <- apply(feat,2,sd)

        return(list('MC' = MCrate, 'AVF'=AVF, 'SDF'=SDF, 'features'=feat, 'rating'=apply(rating
            ,1,table), 'rating.meta'=rating ))

}
```

R-Code 11: IR

```
validate <- function(y.prediction, y.test) {
        return(list('Table'=table(y.test,y.prediction ,dnn=c('Observed', 'Predicted')), 'MCRate
            '=sum(y.prediction==y.test)/length(y.test)))
}
```

R-Code 12: IR

References

Bickel, P. J. and Levina, E. (2004). Some theory for fisher's linear discriminant function, 'naive bayes', and some alternatives when there are many more variables than observations. *Bernoulli*, 10(6):989–1010.

Chang, K.-W., Hsieh, C.-J., and Lin, C.-J. (2008). Coordinate descent method for large-scale l2- loss linear svm. *Journal of Machine Learning Research*, 9:1369–1398.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273–297.

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Fan, J. and Fan, Y. (2008). High-dimensional classification using features annealed independence rules. *The Annals of Statistics*, 36(6):2605–2637.

Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society*, 70(5):849–911.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Friedman, J. (1988). Regularized discriminant analysis. *SLAC - PUB*.

Guo, Y., Hastie, T., and Tibshirani, R. (2005). Regularized discriminant analysis and its application in microarrays. *Manuscript*.

Guo, Y., Hastie, T., and Tibshirani, R. (2012). *Shrunken Centroids Regularized Discriminant Analysis*. http://cran.r-project.org/web/packages/rda/rda.pdf.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *Elements of statistical learning*. Springer Series in Statistics, New York, 2nd edition.

Hastie, T., Tibshirani, R., Narasimhan, B., and Chu, G. (2011). *Pam: prediction analysis for microarrays*. http://cran.r-project.org/web/packages/pamr/pamr.pdf.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for infor- mation storage and organization in the brain. *Psychological Review*, 65:386–408.

Tibshirani, R., Hastie, T., Balasubramanian, N., and Chu, G. (2003). Class prediction by nearest shrunken centroids, with applications to dna microarray. *Statistical Science*, 18(1):104–117.

Tibshirani, R., Hastie, T., Balasubramanian, N., and G., C. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. *PNAS*, 99(10):6567–6572.