ABSTRACT

HWANG, WOOK YEON. Boosting Methods for Variable Selection in High Dimensional Sparse Models. (Under the direction of Dr. Subhashis Ghosal).

First, we propose new variable selection techniques for regression in high dimensional linear models based on a forward selection version of the least absolute selection and shrinkage operator (LASSO), adaptive LASSO or elastic net, respectively to be called as *forward iterative regression and shrinkage technique* (FIRST), adaptive FIRST and elastic FIRST. We exploit the fact that the LASSO, adaptive LASSO and elastic net have closed form solutions when the predictor is one-dimensional. Second, we propose a new variable selection technique for binary classification in high dimensional models based on a forward selection version of the squared support vector machines (SVM) or one-norm SVM, to be called as *forward iterative selection and classification algorithm* (FISCAL). We suggest the squared support vector machines using ℓ_1-norm and ℓ_2-norm simultaneously. The squared support vector machines are convex and differentiable except at zero when the predictor is one-dimensional. We apply the processes to the original one-norm support vector machines. By carefully considering the relationship between estimators at successive stages, we develop fast algorithms to compute our estimators. It is observed that our approaches have better prediction performance for high dimensional sparse models.

Boosting Methods for Variable Selection in High Dimensional Sparse Models

by
Wook Yeon Hwang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina

2009

APPROVED BY:

Dr. Howard Bondell	Dr. Wenbin Lu
Dr. Subhashis Ghosal Chair of Advisory Committee	Dr. Hao Helen Zhang

DEDICATION

I dedicate this dissertation to my wife, Heasin and my son, Eugene.

BIOGRAPHY

Originally, Wook Yeon Hwang majored in Industrial Engineering. He earned a MSE degree in Industrial Engineering from the Arizona State University. At Arizona State University, he worked in the areas of data mining and quality engineering. He won IIE Transactions Best Quality and Reliability Engineering Paper Award in 2008. Before coming to the US, he earned a MS degree from Pohang University of Science and Technology (POSTECH), in Industrial Engineering. At Pohang University of Science and Technology, he specialized in applied statistics and optimization. Then, he worked for 3 years with Korean government agencies.

ACKNOWLEDGMENTS

I truly appreciate Drs. Subhashis Ghoshal, Hao Helen Zhang, Howard Bondell, and Wenbin Lu for their expertise and time. Also, I would like to thank my parents and previous advisors, Drs. George C. Runger and Chi-Hyuck Jun.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Variable selection in predictive models is a major statistical issue in contemporary data analysis because modern data typically involve a lot of predictors, many of which are irrelevant. Such a sparse structure of the predictive function actually allows us to estimate the predictive function fairly accurately even when the number of predictors far exceeds the number of available observations. Removing irrelevant variables from the predictive model is essential since the presence of too many variables may cause overfitting and multicollinearity, which lead to poor prediction of future outcomes. Moreover, the presence of too many variables in the predictive function makes the relation hard to interpret. Since we are typically interested in situations where the number of predictors is much larger than the size of the available sample (large p, small n or LPSN problems), which is also called high dimensional low sample size (HDLSS), we simply restrict ourselves to a linear model. Generally, predictors will be correlated in LPSN problems. In this dissertation, we focus on variable selection for linear regression and binary classification in LPSN problems.

1.1 Variable selection approaches in linear regression

There are several traditional variable selection methods for linear regression. In forward selection, variables are added to the effective subset of predictors sequentially. In backward selection, variables are gradually removed from the collection. In stepwise selection, variables may be either added or removed depending on predictive performance. Typically, to assess performance and effectiveness in variable selection, the mean squared

error (MSE), adjusted R^2 [25], Mallow's C_p [25], prediction sum of squares (PRESS) [25], Akaike information criterion (AIC) [1], Bayesian information criterion (BIC) [27] and others are used. However, methods based on such criteria generally run into difficulty in LPSN problems. Nowadays, regularized linear regression methods such as the non-negative garrote [8], least absolute selection and shrinkage operator (LASSO) [28], adaptive LASSO [35] and elastic net [36] are widely used for variable selection in linear models.

We consider this following linear regression model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{1.1}$$

where $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T \in \mathbb{R}^n$ is a vector of responses, $\boldsymbol{X} = ((X_{ij}))$ is an $n \times p$ matrix for predictors, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ is a vector for parameters, $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^T$ is an n-dimensional random error, whose components are uncorrelated random variables having mean zero and common variance σ^2. Without loss of generality, we assume that the data are centered, so the intercept is not included in the regression model. Throughout this dissertation, we also assume the columns of $\boldsymbol{X}$ are standardized to length 1, i.e., $\sum_{i=1}^n X_{ij}^2 = 1$ for all $j = 1, ..., p$ and $\boldsymbol{Y}$ is centered. Let $\boldsymbol{X}_1^T, \ldots, \boldsymbol{X}_n^T$ stand for the rows of $\boldsymbol{X}$, so that the ith observation may be decomposed as $Y_i = \boldsymbol{X}_i^T\boldsymbol{\beta} + \varepsilon_i$, $i = 1, ..., n$. The model parameter $\boldsymbol{\beta}$ is traditionally estimated by the method of least squares as follows:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \right\}, \tag{1.2}$$

where

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y},$$

provided that the model is of full rank, that is, $\text{rank}(\boldsymbol{X}^T\boldsymbol{X}) = \text{rank}(\boldsymbol{X}) = p$ [25].

However, the variance-covariance matrix of $\hat{\boldsymbol{\beta}}$ given by $\sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ becomes very unstable when components of the predictors are nearly linearly related. In particular, if the number of observations n is less than the number of predictors p, $\boldsymbol{X}^T\boldsymbol{X}$ is necessarily singular so that a generalized inverse matrix should be considered for $\boldsymbol{X}^T\boldsymbol{X}$. In such cases, the least square estimator is not unique, and only certain linear combinations of $\boldsymbol{\beta}$ can be unbiasedly estimated. To stabilize the variability of $\hat{\boldsymbol{\beta}}$, Hoerl and Kennard (1970) proposed

ridge regression estimator which minimizes a penalized sum of squares as follows:

$$\hat{\boldsymbol{\beta}}^R = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}, \tag{1.3}$$

which leads to the solution,

$$\hat{\boldsymbol{\beta}}^R = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_p)^{-1}\boldsymbol{X}^T\boldsymbol{Y},$$

where $\lambda > 0$ is a Lagrangian multiplier regulating the penalty level [22]. The estimator essentially introduces a shrinkage on $\hat{\boldsymbol{\beta}}$ towards zero, and may be viewed as a Bayes estimator with independent identical normal priors on each component of $\boldsymbol{\beta}$.

In LPSN problems, typically many variables are irrelevant, that is, they have true regression coefficient equal to zero. It is essential to filter out insignificant variables in order to reduce the model to a manageable level and reduce prediction errors while estimating the regression coefficients. Thus it is desirable to shrink an insignificant regression coefficient to exactly zero, so that the corresponding variable will be automatically eliminated. In LPSN problem, the non-negative garrote is represented by

$$\boldsymbol{c}^G = \arg\min \left\{ \boldsymbol{c} = (c_1, \ldots, c_p)^T \geq \boldsymbol{0} : \|\boldsymbol{Y} - \boldsymbol{X}(\hat{\boldsymbol{\beta}}^T \circ \boldsymbol{c})\|^2 + \lambda \sum_{j=1}^{p} c_j \right\}, \tag{1.4}$$

where $\circ$ stands for componentwise (Hadamard) multiplication of vectors, λ is a Lagrangian multiplier and non-negative garrote estimator becomes $\hat{\boldsymbol{\beta}}^G = \hat{\boldsymbol{\beta}}^T \circ \boldsymbol{c}^G$ [8]. Under the non-negativity restriction, the second term in the expression above stands for the ℓ_1-norm of the vector $\boldsymbol{c}$. Thus the non-negative garrote essentially looks for a minimizer of sum of squares within the boundary of ℓ_1-balls intersected with the objective function defined by the least square estimator and the vector $\boldsymbol{c}$. The nature of the boundary of ℓ_1-balls implies that the solution will often be located on the boundary, where many of the components are exactly zero. Thus the non-negative garrote shrinks the least square estimator towards zero and sets small coefficients to exactly zero. In order to solve the optimization problem, Breiman (1995) used a modification of the nonnegative least squares algorithm suggested by Lawson and Hanson (1974) [8, 24].

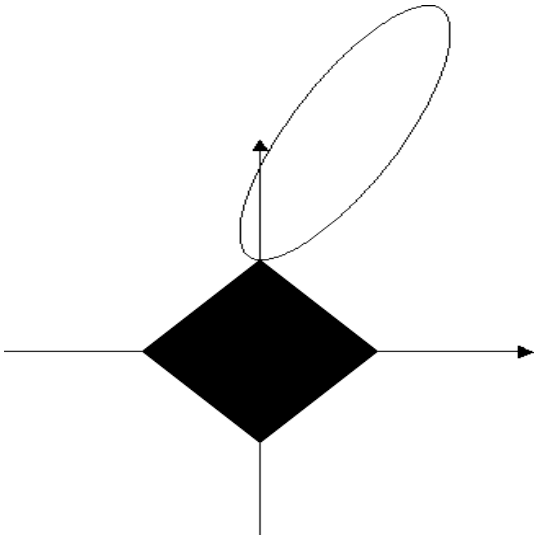

Figure 1.1: LASSO explained graphically

However, the non-negative garotte still depends on the least square estimator in its formulation, which is an inconvenience in LPSN problems. Tibshirani (1996) introduced the least absolute selection and shrinkage operator (LASSO), where like the non-negative garrote, an ℓ_1-penalty is imposed, but the vector $\hat{\boldsymbol{\beta}} \circ \boldsymbol{c}$ is replaced by an arbitrary vector $\boldsymbol{\beta}$ without any non-negativity constraint. In other words, the LASSO estimate is obtained by solving the quadratic optimization problem

$$\hat{\boldsymbol{\beta}}^L = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}, \tag{1.5}$$

where $\lambda > 0$ is a Lagrangian multiplier which is used to regulate the penalty level [28]. That is, the LASSO minimizes the residual sum of squares subject to the lower bounded sum of the absolute value of the coefficients. The nature of the ℓ_1-norm constraint enables the LASSO to make some coefficients exactly zero, which implies that the LASSO can do variable selection in linear models.

The Figure 1.1 (Tibshirani, 1996) explains how the ℓ_1-norm constraint plays a role in making some coefficients exactly zero where the shaded diamond is the ℓ_1-norm constraint and the unshaded oval is the quadratic objective function [28]. On the other hand, the Figure 1.2 (Tibshirani, 1996) explains ridge regression graphically where the shaded circle

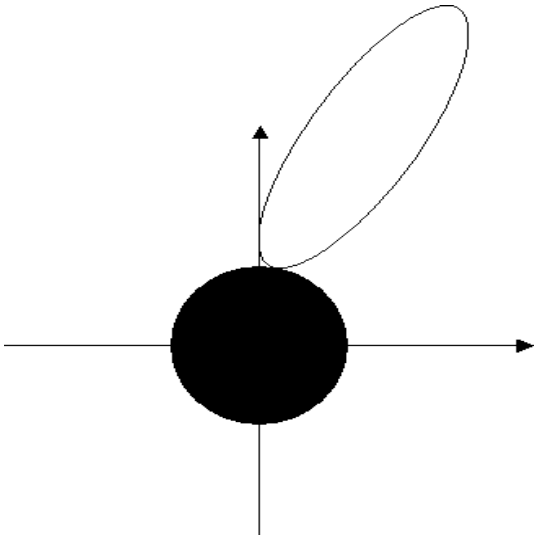

Figure 1.2: Ridge regression explained graphically

is the ℓ_2-norm constraint and the unshaded oval is the quadratic objective function [28]. The one of the LASSO solutions is exactly zero in the Figure 1.1 whereas none of the ridge solutions are not exactly zero in the Figure 1.2, although one of the ridge solutions is close to zero. However, if the two predictors are highly correlated, it may not be desirable to make the one of the two estimates zero. Therefore, the LASSO may not be useful for correlated predictors. In contrast, the ridge penalty may be useful for correlated predictors although ridge regression actually does not do variable selection. Regarding computations, Tibshirani (1996) introduced two algorithms for finding the LASSO solutions [28]. One is based on the ingredients for a procedure which solves the linear least squares problem subject to a general linear inequality constraint where the problem is solved by considering the 2^p inequality constraints sequentially, searching for a feasible solution meeting the Kuhn-Tucker conditions [24]. In the second algorithm, β_j is replaced by $\beta_j^+ - \beta_j^-$ where β_j^+ and β_j^- are nonnegative. He solved a new problem with more variables but fewer constraints. Tibshirani (1996) pointed out that the second algorithm is usually a little faster than the first algorithm [28]. Theoretically, Zhao and Yu (2006) studied model selection consistency of LASSO [33]. We summarize their definitions, conditions and theorem as follows:

1. Equality in Sign: An estimate $\hat{\boldsymbol{\beta}}^n$ is equal in sign with the true model $\boldsymbol{\beta}^n$ which is

written

$$\hat{\boldsymbol{\beta}^n} =_s \boldsymbol{\beta}^n,$$

if and only

$$\text{sign}(\hat{\boldsymbol{\beta}^n}) = \text{sign}(\boldsymbol{\beta}^n).$$

2. Strongly Sign Consistency: If there exists $\lambda_n = f(n)$, that is, a function of n and independent of $\boldsymbol{Y}_n$ or $\boldsymbol{X}_n$ such that

$$\lim_{n\to\infty} P(\hat{\boldsymbol{\beta}^n}(\lambda_n) =_s \boldsymbol{\beta}^n) = 1.$$

3. Strong Irrepresentable Condition: Assume $\frac{1}{n}\boldsymbol{X}_n^T\boldsymbol{X}_n = \boldsymbol{C}^n$,

$$\boldsymbol{\beta}^n_{(1)} = (\beta^n_1, \beta^n_2, ..., \beta^n_q)^T = \{\beta^n_j : \beta^n_j \neq 0\},$$

$$\boldsymbol{\beta}^n_{(2)} = (\beta^n_{q+1}, \beta^n_{q+2}, ..., \beta^n_p)^T = \{\beta^n_j : \beta^n_j = 0\},$$

$$\boldsymbol{C}^n = \begin{pmatrix} \boldsymbol{C}^n_{11} & \boldsymbol{C}^n_{12} \\ \boldsymbol{C}^n_{21} & \boldsymbol{C}^n_{22} \end{pmatrix},$$

where $\boldsymbol{C}_{11}:$ a $q \times q$ matrix, $j = 1, ...p$.

There exists a positive constant vector $\boldsymbol{\eta}$ such that

$$|\boldsymbol{C}^n_{21}(\boldsymbol{C}^n_{11})^{-1} \text{ sign}(\boldsymbol{\beta}^n_{(1)})| \leq \mathbf{1} - \boldsymbol{\eta}.$$

4. Regularity Conditions:

$$\boldsymbol{C}^n \to \boldsymbol{C}, \text{as } n \to \infty,$$

where $\boldsymbol{C}$ is a positive definite matrix. And,

$$\frac{1}{n}\max_{1\leq i\leq n}((\boldsymbol{X}^n_i)^T\boldsymbol{X}^n_i) \to 0, \text{as } n \to \infty.$$

Theorem 1. *For fixed q, p, $\boldsymbol{\beta}^n = \boldsymbol{\beta}$, and $\forall\lambda_n$ that satisfies $\lambda_n/n \to 0$ and $\lambda_n/n^{\frac{1+c}{2}} \to \infty$ with $0 \leq c < 1$, the LASSO is strongly sign consistent under the strong irrepresentable condition and regularity conditions where $\boldsymbol{\beta}^n$ is $\boldsymbol{\beta}$ corresponding to changing sample size.*

While the non-negative garotte and LASSO are very useful for variable selection problems, they do not have closed form expressions in general. In fact, the non-linear optimization problems, (1.4) and (1.5) are quadratic programming problems with linear inequality constraints. In particular, a faster iterative algorithm for computation of the LASSO can be given by a slight modification of the least angle regression (LARS) algorithm introduced by Efron et al. (2004) [14]. The LARS is a less greedy version of the forward stage-wise linear regression. The LARS updates $\hat{\boldsymbol{u}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$ successively. First, $\hat{\boldsymbol{c}} = \boldsymbol{X}^T(\boldsymbol{Y} - \hat{\boldsymbol{u}}_{\mathcal{A}})$ is the vector of current correlation. We define the following notations:

1. $\hat{C} = \max_j\{|\hat{c}_j|\}$.
2. $\mathcal{A} = \{j : |\hat{c}_j| = \hat{C}\}$.
3. $s_j = \text{sign}\{\hat{c}_j\}$ for $j \in \mathcal{A}$.
4. $\boldsymbol{X}_{\mathcal{A}} = (..., s_j\boldsymbol{X}_{\boldsymbol{j}}, ...)$ for $j \in \mathcal{A}$.
5. $\boldsymbol{g}_{\mathcal{A}} = \boldsymbol{X}_{\mathcal{A}}^T\boldsymbol{X}_{\mathcal{A}}$.
6. $\boldsymbol{A}_{\mathcal{A}} = (\mathbf{1}_{\mathcal{A}}^T\boldsymbol{g}_{\mathcal{A}}^{-1}\mathbf{1}_{\mathcal{A}})^{-0.5}$.
7. $\mathbf{1}_{\mathcal{A}}$ = a vector of 1's of length equaling $|\mathcal{A}|$.
8. $\boldsymbol{w}_{\mathcal{A}} = \boldsymbol{A}_{\mathcal{A}}\boldsymbol{g}_{\mathcal{A}}^{-1}\mathbf{1}_{\mathcal{A}}$.
9. $\boldsymbol{u}_{\mathcal{A}} = \boldsymbol{X}_{\mathcal{A}}\boldsymbol{w}_{\mathcal{A}}$.
10. $\boldsymbol{a} = \boldsymbol{X}^T\boldsymbol{u}_{\mathcal{A}}$.
11. $\hat{\gamma} = \min_{j\in\mathcal{A}^c}^{+}\{\frac{\hat{C}-\hat{c}_j}{A_{\mathcal{A}}-a_j}, \frac{\hat{C}+\hat{c}_j}{A_{\mathcal{A}}+a_j}\}$.
12. $\hat{\boldsymbol{d}}$ = the p-vector equaling $s_j w_{\mathcal{A}_j}$ for $j \in \mathcal{A}$.
13. $\gamma_j = -\hat{\beta}_j/\hat{d}_j$ for $j \in \mathcal{A}$.
14. $\tilde{\gamma} = \min_{\gamma_j>0}\ \gamma_j$ for $j \in \mathcal{A}$,

where $\min^+$ means that we choose minimum over only positive components. Eventually, we can update the equiangular vector, $\hat{\boldsymbol{u}}_{\mathcal{A}}$,

$$\hat{\boldsymbol{u}}_{\mathcal{A}+} = \hat{\boldsymbol{u}}_{\mathcal{A}} + \hat{\gamma}\boldsymbol{u}_{\mathcal{A}}. \tag{1.6}$$

A simple modification of LARS gives an algorithm for the LASSO. We see that

$$\boldsymbol{u}(\gamma) = \boldsymbol{X}\boldsymbol{\beta}(\gamma), \tag{1.7}$$

where

$$\beta_j(\gamma) = \hat{\beta}_j + \gamma \hat{d}_j,$$

for $j \in \mathcal{A}$. For LASSO modification, if $\tilde{\gamma} < \hat{\gamma}$, stop the LARS step at $\gamma = \tilde{\gamma}$ and $\tilde{j}$ is not considered for the next equiangular direction. That is, $\mathcal{A}+ = \mathcal{A} - \{\tilde{j}\}$. Then the LARS estimates give the LASSO estimates. However, in applications where both p and n are large, the LARS algorithm may take a long time to compute.

Intuitively, it is preferable to penalize different components differently in LASSO in tune with the size of some estimate of their regression coefficients, leading to the concept of adaptive LASSO [35]. The adaptive LASSO can be described as the solution of the quadratic optimization problem

$$\hat{\boldsymbol{\beta}}^{AL} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} \frac{|\beta_j|}{|\tilde{\beta}_j|^{\gamma}} \right\}, \quad \lambda > 0,\ \gamma > 0, \tag{1.8}$$

where $\tilde{\beta}_j$ is an initial $\sqrt{n}$ consistent estimator of β_j, $j = 1, \ldots, p$. Indeed, it can be shown that such estimators can possess important oracle properties [35]. First, we assume the following conditions:

1. $\frac{1}{n}\boldsymbol{X}^T\boldsymbol{X} \to \boldsymbol{C}$ where $\boldsymbol{C}$ is a positive definite matrix.
2. $\mathcal{A} = \{1, 2, ..., q\} = \{j : \beta_j \neq 0\}$.
3. $$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{pmatrix},$$
   where $\boldsymbol{C}_{11}$: a $q \times q$ matrix.

If $\lambda_n\sqrt{n} \to 0$ and $\lambda_n n^{(\gamma-1)/2} \to \infty$, the adaptive LASSO estimates satisfies
a. Consistency in variable selection : $\lim_{n\to\infty} P(\mathcal{A}_n = \mathcal{A}) = 1$ where $\mathcal{A}_n$ is $\mathcal{A}$ corresponding to changing sample size.

b. Asymptotic normality : $\sqrt{n}(\boldsymbol{\beta}_{\mathcal{A}}^{AL(n)} - \boldsymbol{\beta}_{\mathcal{A}}) \rightarrow_d N(0, \sigma^2 \cdot \boldsymbol{C}_{11}^{-1})$.

The least square estimator of $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}$ is a natural choice for $\tilde{\boldsymbol{\beta}}$, but the choice suffers in LPSN problems where it is not unique because of singularity. The ridge regression estimator is a more sensible choice for $\tilde{\boldsymbol{\beta}}$ [23]. Usually, one chooses $\gamma = 1$ in (1.8) where it was proven that the adaptive LASSO with additional sign constraints is the same as the non-negative garotte [35]. Regarding computations, the LARS algorithm can be easily adapted for solving the adaptive LASSO [35].

When the regressors are highly correlated, the LASSO sometimes shows some erratic behaviors in selecting variables arbitrarily from a group of correlated variables, and in reversing the sign of the estimate of a regression coefficient as the smoothing parameter varies. To avoid these shortcomings, Zou and Hastie (2005) suggested imposing a quadratic penalty in addition to LASSO's ℓ_1-penalty and called the resulting procedure an elastic net (EN) [36]. Thus the naive elastic net estimator with smoothing parameter $\lambda_1 > 0$ and $\lambda_2 > 0$ is defined as

$$\hat{\boldsymbol{\beta}}^{EN} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{1.9}$$

If λ_1 is zero, the EN becomes the ridge regression. Similarly, if λ_2 is zero, the EN becomes the LASSO. The Figure 1.3 (Bondell and Reich, 2008) shows that constraint region in the EN approaches from constraint region for the LASSO to constraint region for the ridge regression [6]. Regarding computations, it was proven that the naive elastic net is equivalent to a LASSO problem on an augmented data [36]. Therefore the LARS algorithm can be used to locate the entire elastic net solution path which is called the LARS-EN algorithm.

Eventually, Zou and Zhang (2009) suggested the adaptive elastic net designed for high-dimensional data analysis by applying the adaptive LASSO concept to the EN as follows:

$$\hat{\boldsymbol{\beta}}^{AEN} = (1 + \frac{\lambda_2}{n}) \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_1 \sum_{j=1}^{p} \hat{w}_j |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\}, \tag{1.10}$$

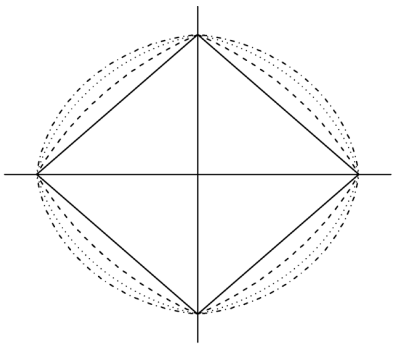

Figure 1.3: Constraint region for the LASSO (solid line), along with three choices of tuning parameter for the Elastic Net

where γ is a positive parameter,

$$\hat{w}_j = (|\hat{\beta}_j^{EN}|)^{-\gamma},$$

$$\hat{\boldsymbol{\beta}}^{EN} = (1 + \frac{\lambda_2}{n}) \arg\min_{\boldsymbol{\beta}} \{\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2\}$$

which is the corrected elastic net estimate [36, 37]. Especially, they studied its asymptotic properties under the assumption that the dimension diverges with the sample size [37].

Additionally, octagonal shrinkage and clustering algorithm for regression (OSCAR) was developed for a group of correlated variables [6]. Bondell and Reich (2008) combined sum of pairwise absolute maximum with LASSO's ℓ_1-penalty. The OSCAR estimator with smoothing parameter $\lambda > 0$ and $c > 0$ is represented as

$$\hat{\boldsymbol{\beta}}^{O} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda [\sum_{j=1}^{p} |\beta_j| + c \sum_{j<k}^{p} \max\{|\beta_j|, |\beta_k|\}] \right\}. \tag{1.11}$$

If c is zero, the OSCAR becomes the LASSO. The Figure 1.4 (Bondell and Reich, 2008) shows that constraint region in the OSCAR approaches from constraint region for the LASSO to a square-shaped constraint region [6]. They converted (1.11) into a quadratic

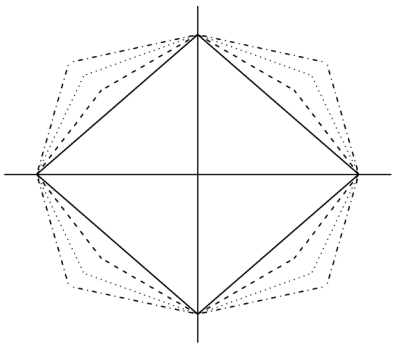

Figure 1.4: Constraint region for the LASSO (solid line), along with three choices of tuning parameter for the OSCAR

programming problem with extremely sparse constraints after writing $\beta_j = \beta_j^+ - \beta_j^-$ with both β_j^+ and β_j^- being non-negative. Then they applied SQOPT (Gill et al., 2005) designed for large-scale problems with sparse matrices to the quadratic programming problem [20].

One problem in the LASSO is that the one-norm penalty puts a steep penalty to important coefficients. As a result, Fan and Li (2001) suggested the smoothly clipped absolute deviation (SCAD) as follows:

$$\hat{\boldsymbol{\beta}}^S = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} p_\lambda(|\beta_j|) \right\}, \tag{1.12}$$

where $a > 0$, $\lambda > 0$, and $p_\lambda(|\beta_j|)$ is given by its derivative,

$$p'_\lambda(|\beta_j|) = \lambda\{I(|\beta_j| \leq \lambda) + \frac{(a\lambda - |\beta_j|)_+}{(a-1)\lambda} I(|\beta_j| > \lambda)\}, \tag{1.13}$$

where $a > 2$ and $\beta > 0$ [15]. The SCAD penalty, $p_\lambda(|\beta_j|)$ puts relatively small penalty to important coefficients. The SCAD estimators also enjoy important properties as follows. If $\lambda_n \to 0$ and $\sqrt{n}\lambda_n \to \infty$ as $n \to \infty$, then with probability tending to 1, a $\sqrt{n}$ consistent penalized likelihood estimator, $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\beta}}_{\mathbf{1}}, \hat{\boldsymbol{\beta}}_{\mathbf{2}})^T$ satisfies

1. sparsity : $\hat{\boldsymbol{\beta}}_{\mathbf{2}} = 0$ where $\boldsymbol{\beta}_{\mathbf{2}} = 0$.

2. asymptotic normality under the specific conditions.

They suggested a new unified algorithm to solve (1.12) via local quadratic approximations [15]. In particular, they showed that the second term in (1.12) can be locally approximated by a quadratic function. Then they applied the Newton-Raphson algorithm.

In addition to the LARS algorithm, other iterative approaches have been used for variable selection in the high dimensional linear models. Basically, forward stage-wise linear regression and L_2Boosting with componentwise linear least squares decompose multiple regression into simple regression sequentially despite that there are some differences between them. Most significantly, shrinkage occurs in the updating stage of L_2Boosting not in that of forward stage-wise linear regression. Also, a maximum number of iterations is chosen based on the $\mathrm{AIC_C}(m) = \log(\hat{\sigma}^2) + \frac{1+\mathrm{trace}(\mathcal{B}_m)/n}{1-(\mathrm{trace}(\mathcal{B}_m)+2)/n}$ explained in the followed L_2Boosting description whereas it is given arbitrarily in forward stage-wise linear regression. Similarly in the LASSO, a modification of the LARS algorithm becomes forward stage-wise linear regression [14]. Forward stage-wise linear regression is described as follows [21].

Fix a big number of iteration steps, say M to be determined by some well defined method. Set $\nu > 0$ to some small constant. Set initial $Y_{i,1} = Y_i$ for $i = 1, \ldots, n$ and initial prediction vector $\hat{\boldsymbol{f}}_1 = \mathbf{0}$.

For $m = 1, \ldots, M$, repeat

1. Standardization step: Replace X_{ij} by $(X_{ij} - \bar{X}_j)/\{\sum_{i=1}^n (X_{ij} - \bar{X}_j)^2\}^{1/2}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, where $\bar{X}_j = n^{-1} \sum_{i=1}^n X_{ij}$. Thus, from now onwards, we shall assume that $\bar{X}_j = 0$ and $\sum_{i=1}^n X_{ij}^2 = 1$ for all $j = 1, \ldots, p$.

2. Regression step: Calculate ordinary least square estimates, $\hat{\beta}_{1,m}, \ldots, \hat{\beta}_{p,m}$, by $\hat{\beta}_{j,m} = \sum_{i=1}^n Y_{i,m} X_{ij}$, $j = 1, \ldots, p$.

3. Selection step: Select $X_{j_m^*}$ as the most effective variable in the mth iteration step, where $j_m^* = \arg_j \min \sum_{i=1}^n (Y_{i,m} - X_{ij}\hat{\beta}_{j,m})^2$, $j = 1, \ldots, p$.

4. Updating stage: Update m to $m+1$, $Y_{i,m}$ to $Y_{i,m+1} = Y_{i,m} - X_{ij_m^*}(\nu \cdot \mathrm{sign}(\hat{\beta}_{j_m^*,m}))$, and $\hat{\boldsymbol{f}}_m$ to $\hat{\boldsymbol{f}}_{m+1} = \hat{\boldsymbol{f}}_m + X_{j_m^*}(\nu \cdot \mathrm{sign}(\hat{\beta}_{j_m^*,m}))$.

Also, L_2Boosting with componentwise linear least squares is described as follows [9].

Fix a maximum number of iteration steps, say M=arg$_{1\leq m\leq m^*}$ min AIC$_{\mathrm{C}}(m)$ where m^* is a upper bound. Set initial $Y_{i,1} = Y_i$ for $i = 1, \ldots, n$, $\hat{\boldsymbol{f}}_1 = \mathbf{0}$ and $0 < \nu \leq 1$.

For $m = 1, \ldots, M$, repeat

1. Standardization step: Replace X_{ij} by $(X_{ij} - \bar{X}_j)/\{\sum_{i=1}^n (X_{ij} - \bar{X}_j)^2\}^{1/2}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, where $\bar{X}_j = n^{-1} \sum_{i=1}^n X_{ij}$. Thus, from now onwards, we shall assume that $\bar{X}_j = 0$ and $\sum_{i=1}^n X_{ij}^2 = 1$ for all $j = 1, \ldots, p$.

2. Regression step: Calculate ordinary least square estimates, $\hat{\beta}_{1,m}, \ldots, \hat{\beta}_{p,m}$, by $\hat{\beta}_{j,m} = \sum_{i=1}^n Y_{i,m} X_{ij}$, $j = 1, \ldots, p$.

3. Selection step: Select $X_{j_m^*}$ as the most effective variable in the mth iteration step, where $j_m^* = \arg_j \min \sum_{i=1}^n (Y_{i,m} - X_{ij}\hat{\beta}_{j,m})^2$, $j = 1, \ldots, p$.

4. Updating stage: Update m to $m + 1$, $Y_{i,m}$ to $Y_{i,m+1} = Y_{i,m} - \nu X_{ij_m^*} \hat{\beta}_{j_m^*,m}$, and $\hat{\boldsymbol{f}}_m$ to $\hat{\boldsymbol{f}}_{m+1} = \hat{\boldsymbol{f}}_m + \nu X_{j_m^*} \hat{\beta}_{j_m^*,m}$.

For a stopping rule, L_2Boosting uses AIC$_{\mathrm{C}}$ criterion [9, 11]. Here, the degrees of freedom for boosting are denoted by

$$\mathcal{H}_{(j)} = X_j {X_j}^T / \|X_j\|^2, j = 1, ..., p \text{ where } X_j = (X_{1j}, ..., X_{nj})^T.$$

The L_2Boosting hat-matrix is defined as

$$\mathcal{B}_m = I - (I - \nu \mathcal{H}_{\hat{s}_1}) ... (I - \nu \mathcal{H}_{\hat{s}_m}),$$

where $\hat{s}_i \in \{1, ...d\}$ denotes the component which is selected in the ith boosting iteration. Then we can use a corrected version of AIC$_{\mathrm{C}}$ as follows:

$$\mathrm{AIC}_{\mathrm{C}}(m) = \log(\hat{\sigma}^2) + \frac{1 + \mathrm{trace}(\mathcal{B}_m)/n}{1 - (\mathrm{trace}(\mathcal{B}_m) + 2)/n},$$

$$\hat{\sigma}^2 = \sum_{i=1}^n (Y_i - (\mathcal{B}_m \boldsymbol{Y})_i)^2 / n,$$

$$\boldsymbol{Y} = (Y_1, ..., Y_n)^T.$$

Finally, we choose M to give minimum $\mathrm{AIC_C}(m)$. Theoretically, they proved the consistency of the L_2Boosting prediction estimates under limited assumptions as follows.

$$\mathrm{E}_{\boldsymbol{X}} |\hat{\boldsymbol{f}}_{m_n} - \boldsymbol{f}_n|^2 = o_P(1) \text{ as } n \to \infty,$$

where the following conditions are given:

1. $Y_i = \boldsymbol{f}_n + \varepsilon_i$, $i = 1, ..., n$.
2. $\boldsymbol{f}_n = \sum_{j=1}^{p_n} \beta_{j,n} X^{(j)}$, where the number of predictors, p_n is allowed to grow with sample size n.
3. For some sequence $(m_n)_{n\in\mathbb{N}}$, which is allowed to be random and depending on the realizations of the data, with $(m_n) = o_P(n^{\xi/4})$ as $n \to \infty$ where $0 < \xi < 1$.
4. The symbol, $\boldsymbol{X}$ denotes a new predictor variable, independent of and with the same distribution as the data.

It is well-known that the LARS procedure provides a state-of-the-art algorithm to compute the entire solution path for LASSO-type problems. Recently, Friedman et al. (2007) proposed another iterative algorithm, the coordinate-wise descent algorithm (CDA) for solving the regularized regression problems such as the LASSO and EN [17]. The main idea of the CDA is to successively minimize the objective function with respect to one parameter at a time, while holding the remaining parameters at their current values and updating residuals successively. We use the LASSO to describe the algorithm. At each step, the coordinate-wise descent algorithm solves

$$\arg\min_{\beta_j} \left\{ \sum_{i=1}^{n} (Y_i - \sum_{k\neq j} X_{ik}\tilde{\beta}_k - X_{ij}\beta_j)^2 + \lambda \sum_{k\neq j} |\tilde{\beta}_k| + \lambda|\beta_j| \right\}, \tag{1.14}$$

where the minimization is with respect to β_j and all the remaining parameters β_k for $k \neq j$ are fixed at their current values $\tilde{\beta}_k$. With one predictor, the lasso solution to (1.14) can be defined as follows:

$$\hat{\beta}_j^L = \begin{cases} \hat{\beta}_j - \frac{\lambda}{2}, & \text{if } \hat{\beta}_j \geq \lambda/2, \\ 0, & \text{if } |\hat{\beta}_j| < \lambda/2, \\ \hat{\beta}_j + \frac{\lambda}{2}, & \text{if } \hat{\beta}_j \leq -\lambda/2, \end{cases} \tag{1.15}$$

where

$$\hat{\beta}_j = \sum_{i=1}^{n} X_{ij}(Y_i - \sum_{k \neq j} X_{ik}\tilde{\beta}_k) \text{ [17].}$$

Similarly, the CDA for the elastic net can solve the following problem at each step:

$$\arg\min_{\beta_j} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (Y_i - \sum_{k \neq j} X_{ik}\tilde{\beta}_k - X_{ij}\beta_j)^2 + \lambda P_\alpha(\boldsymbol{\beta}) \right\}, \tag{1.16}$$

where

$$P_\alpha(\boldsymbol{\beta}) = \sum_{j=1}^{p} [\alpha|\beta_j| + \frac{1}{2}(1-\alpha)\beta_j^2], 0 \leq \alpha \leq 1 \text{ [18].}$$

The solution to (1.16) can be defined as follows:

$$\hat{\beta}^{EN} = \begin{cases} \frac{\frac{1}{n}\hat{\beta} - \lambda\alpha}{1+\lambda(1-\alpha)}, & \text{if } \frac{1}{n}\hat{\beta} \geq \lambda\alpha, \\ 0, & \text{if } |\frac{1}{n}\hat{\beta}| < \lambda\alpha, \\ \frac{\frac{1}{n}\hat{\beta} + \lambda\alpha}{1+\lambda(1-\alpha)}, & \text{if } \frac{1}{n}\hat{\beta} \leq -\lambda\alpha. \end{cases} \tag{1.17}$$

Actually, the optimization for obtaining the LASSO and EN solutions using only one variable is repeated for $j = 1, 2, ..., p, 1, 2, ...$ until convergence. Friedman et al. (2007) pointed out that the CDA converges to the optimal solution and gave very competitive performance with the LARS procedure especially when p is large [17]. Above all, it is known that the CDA is much faster than the LARS.

1.2 Variable selection approaches in binary classification

A major issue in supervised learning is to identify members of two different classes which is called binary classification. We can use the traditional logistic regression model,

$$\log(\frac{p_i}{1-p_i}) = \boldsymbol{X_i}^T\boldsymbol{\beta}, i = 1, .., n,$$

where Y_i is a 0/1 response, X_i^T is a $1 \times p$ vector for predictors, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ is a vector for parameters for binary classification and

$$\mathrm{E}(Y_i|\boldsymbol{X}_i) = p_i = \frac{\exp(\boldsymbol{X_i}^T\boldsymbol{\beta})}{1+\exp(\boldsymbol{X_i}^T\boldsymbol{\beta})}.$$

The log-likelihood can be written by

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \{Y_i \log(p_i) + (1 - Y_i) \log(1 - p_i)\}. \tag{1.18}$$

The Newton-Raphson algorithm solves the derivative equations in order to estimate the parameters iteratively. However, the logistic regression model struggles with curse of dimensionality in LPSN problems. As a result, the penalized logistic regression model can be considered for LPSN problems instead of the original logistic regression model. The log-likelihood for the penalized logistic regression model can be modified as follows:

$$l(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \{Y_i \log(p_i) + (1 - Y_i) \log(1 - p_i)\} - \lambda P_\alpha(\boldsymbol{\beta}), \tag{1.19}$$

where

$$P_\alpha(\boldsymbol{\beta}) = \sum_{j=1}^{p} [\alpha|\beta_j| + \frac{1}{2}(1 - \alpha)\beta_j^2], 0 \leq \alpha \leq 1 \text{ [18]}.$$

The log-likelihood part of (1.19) becomes the unpenalized concave function of the parameters. The Newton-Raphson algorithm for maximizing the unpenalized concave function is equivalent to iteratively reweighted least squares [18]. They formed a quadratic approximation to the negative unpenalized log-likelihood using Taylor expansion about current estimates. Finally, they applied the coordinate descent algorithm (CDA) on the penalized weighted-least-squares problem.

In binary classification, training data, $(\boldsymbol{X_1}, Y_1), ..., (\boldsymbol{X_n}, Y_n) \in \mathbb{R}^p \times \{-1, 1\}$ are used for estimating a function $f : \mathbb{R}^p \to \{-1, 1\}$. The estimated function $\hat{f} : \mathbb{R}^p \to \{-1, 1\}$ will classify a new test data, $\boldsymbol{X} \in \mathbb{R}^p$ into one of the categories -1 and $+1$. Here, we can define misclassification training error,

$$\frac{1}{2n} \sum_{i=1}^{n} |f(\boldsymbol{X_i}) - Y_i|,$$

and misclassification test error,

$$\frac{1}{2} \int |f(\boldsymbol{X}) - Y| dP(\boldsymbol{X}, Y),$$

where P is the joint distribution of a future observation $(\boldsymbol{X}, Y)$. The technique of support vector machines (SVM), which maximizes the distance between the classified points and the

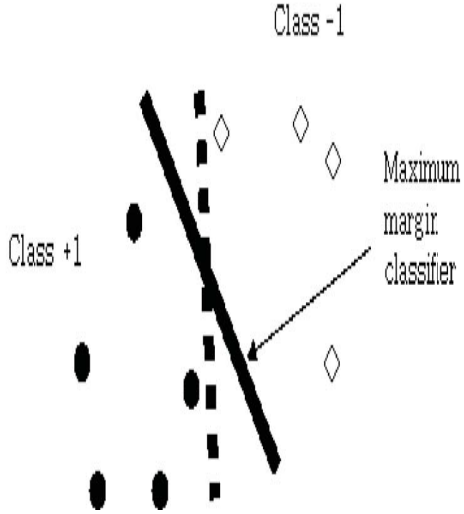

Figure 1.5: Maximum margin classifier in a linearly separable case

classification boundary line has been widely used for binary classification. Since the SVM is a maximum margin classifier, small perturbations of $\boldsymbol{X_1}, ..., \boldsymbol{X_n} \in \mathbb{R}^p$ may not affect misclassification errors [4]. The maximum margin classifier described in Figure 1.5 divides both classes farthest which leads to improvement in misclassification errors. In addition, the Vapnik-Chervonenkis (VC) theory gives bounds on the test error [26]. Most significantly, Bennett and Bredensteiner (2000) mentioned that the test error bounds are minimized by maximizing the margin [5]. Then we expect better test error. As a result, SVM become a reasonable binary classifier.

The SVM originated from mathematical models based on geometric principles for binary classification, where both linearly separable cases and linearly inseparable cases were considered [4]. In this dissertation, if a linear classifier separates the two classes without errors, it is called linearly separable. Otherwise it is called inseparable. Generally, training data, $(\boldsymbol{X_1}, Y_1), ..., (\boldsymbol{X_n}, Y_n) \in \mathbb{R}^p \times \{-1, 1\}$ fall in the linearly inseparable case. The SVM for linearly inseparable cases is obtained by

$$\min_{\boldsymbol{\beta}, \beta_0, \epsilon_1, ..., \epsilon_n} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum_{i=1}^{n} \epsilon_i, \tag{1.20}$$

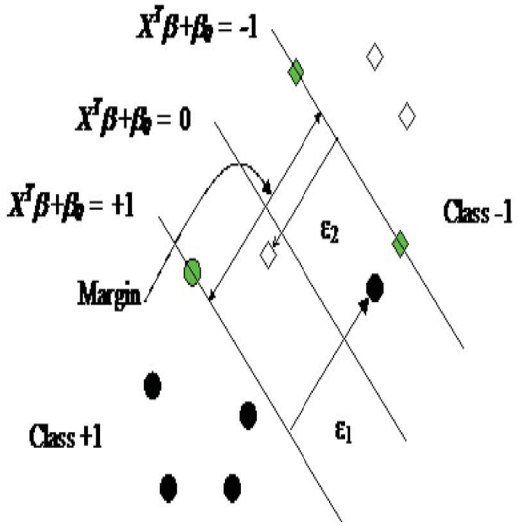

Figure 1.6: Support vector machines in a linearly inseparable case

subject to

$$Y_i(\boldsymbol{X_i}^T\boldsymbol{\beta} + \beta_0) \geq 1 - \epsilon_i,\ \epsilon_i \geq 0\ \forall_i, \gamma > 0,$$

where

$$f(\boldsymbol{X}) = \text{sign}(\boldsymbol{X}^T\boldsymbol{\beta} + \beta_0)\ [21].$$

Figure 1.6 explains (1.20) well. All the points in class $+1$ and -1 are constrained by the two parallel supporting planes,

$$\boldsymbol{X_i}^T\boldsymbol{\beta} + \beta_0 = +1,$$

$$\boldsymbol{X_i}^T\boldsymbol{\beta} + \beta_0 = -1,$$

where an overlap of the two classes is allowed. This is quantified by ϵ_i's, which are called the slack variables. In particular, the distance between the supporting planes, called the margin, is $\frac{2}{\|\boldsymbol{\beta}\|}$. In fact, we minimize the objective function, $\frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma\sum_{i=1}^n \epsilon_i$ subject to the constraints in (1.20). That is, the SVM controls the trade-off between the margin and overlap. The dual of the (1.20) is a quadratic programming problem which can be derived by using Lagrangian multipliers. By solving the dual, we can estimate $\boldsymbol{\beta}$. The Lagrangian function is

$$L(\boldsymbol{\beta}, \beta_0, \epsilon_i, \alpha_i, u_i) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma\sum_{i=1}^n \epsilon_i - \sum_{i=1}^n \alpha_i(Y_i(\boldsymbol{X_i}^T\boldsymbol{\beta} + \beta_0) - 1 + \epsilon_i) - \sum_{i=1}^n u_i\epsilon_i, \quad (1.21)$$

where $\alpha_i \geq 0, u_i \geq 0$.

The derivatives of the Lagrangian function are

$$\frac{\partial}{\partial \beta_0} L(\boldsymbol{\beta}, \beta_0, \epsilon_i, \alpha_i, u_i) = 0, \tag{1.22}$$

$$\frac{\partial}{\partial \boldsymbol{\beta}} L(\boldsymbol{\beta}, \beta_0, \epsilon_i, \alpha_i, u_i) = 0, \tag{1.23}$$

$$\frac{\partial}{\partial \epsilon_i} L(\boldsymbol{\beta}, \beta_0, \epsilon_i, \alpha_i, u_i) = 0. \tag{1.24}$$

These equations become

$$\sum_{i=1}^{n} Y_i \alpha_i = 0, \tag{1.25}$$

$$\boldsymbol{\beta} = \sum_{i=1}^{p} Y_i \alpha_i \boldsymbol{X_i}, \tag{1.26}$$

$$\alpha_i = \gamma - u_i, \forall_i. \tag{1.27}$$

By plugging the two equations, (1.25) and (1.26) into the Lagrangian function given by (1.21), we can obtain the dual of (1.20) as follows:

$$\min_{\alpha_1,\ldots,\alpha_n} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} Y_i Y_j \alpha_i \alpha_j \boldsymbol{X_i}^T \boldsymbol{X_j} - \sum_{i=1}^{n} \alpha_i, \tag{1.28}$$

subject to

$$\sum_{i=1}^{n} Y_i \alpha_i = 0,\ 0 \leq \alpha_i\ \leq \gamma,\ \forall_i \text{ [21]}.$$

The Karush-Kuhn-Tucker conditions are given by

$$\alpha_i \{ Y_i (\boldsymbol{X_i}^T \boldsymbol{\beta}) + \beta_0 - 1 + \epsilon_i \} = 0, \tag{1.29}$$

$$\epsilon_i (\gamma - \alpha_i) = 0,\ \forall_i \text{ [2]}. \tag{1.30}$$

This implies that non-zero α_i where $0 \leq \alpha_i \leq \gamma$, are on one of the two parallel planes shown in Figure 1.6. In particular, a non-zero α_i is called a support vector. We can estimate the slope of the planes by

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^{p} Y_i \hat{\alpha}_i \boldsymbol{X_i},$$

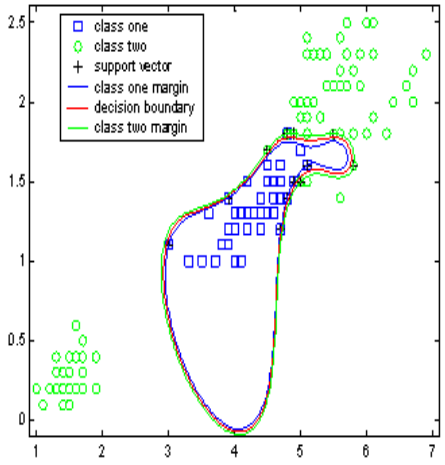

Figure 1.7: Non-linear SVM

and the threshold of the planes, β_0 with the support vectors where $0 < \alpha_i \leq \gamma$. Besides, we can consider non-linear SVM by considering a basis function, $h(\boldsymbol{X})$ instead of $\boldsymbol{X}$, which produces $f(\boldsymbol{X}) = \text{sign}(h(\boldsymbol{X})^T\boldsymbol{\beta} + \beta_0)$. Most significantly, the dual changes into

$$\min_{\alpha_1,\ldots,\alpha_n} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} Y_iY_j\alpha_i\alpha_j\langle h(\boldsymbol{X_i}), h(\boldsymbol{X_j})\rangle - \sum_{i=1}^{n}\alpha_i, \tag{1.31}$$

subject to

$$\sum_{i=1}^{n} Y_i\alpha_i = 0,\ 0 \leq \alpha_i \leq \gamma,\ \forall_i,$$

where $\langle h(\boldsymbol{X_i}), h(\boldsymbol{X_j})\rangle$ may be chosen to be $\exp\{\|\boldsymbol{X_i} - \boldsymbol{X_j}\|^2\}$. The example of non-linear SVM using the kernel function, $\exp\{\|\boldsymbol{X_i} - \boldsymbol{X_j}\|^2\}$ can be shown in Figure 1.7.

On the other hand, the one-norm support vector machines (SVM) are obtained by replacing $\|\boldsymbol{\beta}\|^2$ with $\sum_{j=1}^{p}|\beta_j|$ in (1.20) as follows:

$$\min_{\boldsymbol{\beta},\beta_0,\epsilon_1,\ldots,\epsilon_n,\boldsymbol{s}} \frac{1}{2}\boldsymbol{e}^T\boldsymbol{s} + \gamma\sum_{i=1}^{n}\epsilon_i, \tag{1.32}$$

subject to

$$Y_i(\boldsymbol{X_i}^T\boldsymbol{\beta} + \beta_0) \geq 1 - \epsilon_i,\ \forall_i,$$

$$-\boldsymbol{s} \leq \boldsymbol{\beta} \leq \boldsymbol{s},$$

$$\boldsymbol{s} \geq \boldsymbol{0},\ \epsilon_i \geq 0,\ \forall_i,$$

where $\gamma > 0,\ \boldsymbol{e} = (1, ..., 1)^T \in \mathbb{R}^p, \boldsymbol{s} = (s_1, ..., s_p) \in \mathbb{R}^p$ [4]. One of the main advantages of the one-norm SVM is that they can be solved by linear programming. Moreover, the one-norm SVM perform automatic variable selection simultaneously in classification problems like the LASSO for linear regression. Bradley and Magasarian (1998) pointed out that we use the ℓ_∞-norm to measure the distance between the two planes, $\frac{2}{\|\boldsymbol{\beta}\|_\infty}$ in the one-norm SVM, whereas in the original two-norm SVM, the distance between the two planes is $\frac{2}{\|\boldsymbol{\beta}\|_2}$ [7]. Most of all, they showed in the studies that the one-norm SVM exhibits strong variable selection.

For a more efficient and faster computation algorithm to solve one-norm SVM, Fung and Mangasarian (2004) suggested Newton method for linear programming SVM (NLPSVM) [19]. They considered the following optimization problem for the one-norm SVM:

$$\min_{\boldsymbol{p},\boldsymbol{q},\gamma,\boldsymbol{Y}} \nu \boldsymbol{e}^T \boldsymbol{Y} + \boldsymbol{e}^T(\boldsymbol{p} + \boldsymbol{q}), \tag{1.33}$$

subject to

$$\boldsymbol{D}(\boldsymbol{A}(\boldsymbol{p} - \boldsymbol{q}) - \boldsymbol{e}\gamma) + \boldsymbol{Y} \geq \boldsymbol{e},$$

where $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{Y} \geq \boldsymbol{0},\ \boldsymbol{\beta} = \boldsymbol{p} - \boldsymbol{q}$. Besides, $\boldsymbol{D}$ is a $n \times n$ diagonal matrix with ones or minus ones along with its diagonal, $\boldsymbol{A} \in \mathbb{R}^{n \times p}$ is a matrix to represent predictors and $\nu > 0$ is a shrinkage parameter. The dual of the linear program is the following:

$$\min_{\boldsymbol{u} \in \mathbb{R}^n} \boldsymbol{e}^T \boldsymbol{u}, \tag{1.34}$$

subject to

$$-\boldsymbol{e} \leq \boldsymbol{A}^T \boldsymbol{D} \boldsymbol{u} \leq \boldsymbol{e},$$

$$-\boldsymbol{e}^T \boldsymbol{D} \boldsymbol{u} = \boldsymbol{0},$$

$$\boldsymbol{u} \leq \nu \boldsymbol{e},$$

$$\boldsymbol{u} \geq \boldsymbol{0}.$$

The asymptotic exterior penalty problem for this linear program is the nonnegatively constrained minimization problem below which is a convex penalty function:

$$\min -\epsilon \boldsymbol{e}^T \boldsymbol{u} + \frac{1}{2}\|(\boldsymbol{A}^T \boldsymbol{D}\boldsymbol{u} - \boldsymbol{e}) + \|^2 + \frac{1}{2}\| - (\boldsymbol{A}^T \boldsymbol{D}\boldsymbol{u} - \boldsymbol{e})_+\|^2 + \frac{1}{2}\|\boldsymbol{e}^T \boldsymbol{D}\boldsymbol{u}\|^2 \tag{1.35}$$
$$+\frac{1}{2}\|\boldsymbol{e}^T \boldsymbol{D}\boldsymbol{u}\|^2 + \frac{1}{2}\|(\boldsymbol{u} - \nu \boldsymbol{e})_+\|^2 + \frac{\alpha}{2}\|(\boldsymbol{u})_+\|^2,$$

where ϵ is a positive penalty parameter that needs to approach zero for standard penalty application for solving the dual linear system [3, 13]. In particular, they used the NLPSVM in order to solve the asymptotic exterior penalty problem for the one-norm SVM [19].

Furthermore, Zhang et al. (2006) realized that the one-norm penalty in one-norm SVM applies a steep penalty to big coefficients as shown in the Figure 1.8, whereas the SCAD penalty is the same as the one-norm penalty for small coefficients as well as applies a constant penalty to big coefficients [32]. As a result, Zhang et al. (2006) suggested the SCAD SVM by replacing one-norm penalty with the SCAD penalty in the one-norm SVM in order to prevent such a disadvantage of the one-norm penalty [32]. The SCAD penalty function is given by

$$p_\lambda(\beta) = \begin{cases} \lambda|\beta|, & \text{if } |\beta| \leq \lambda, \\ -\frac{(|\beta|^2 - 2a\lambda|\beta| + \lambda^2)}{2(a-1)}, & \text{if } \lambda < |\beta| \leq a\lambda, \\ \frac{(a+1)\lambda^2}{2}, & \text{if } |\beta| > a\lambda, \end{cases} \tag{1.36}$$

whose example is shown in Figure 1.9 [15, 32]. Particularly, we can represent (1.20) with another optimization problem. When $\frac{1}{2\gamma} = \lambda$, (1.20) can be simplified to

$$\arg\min_{\boldsymbol{\beta}, \beta_0} \left\{ \sum_{i=1}^{n} [1 - Y_i(\boldsymbol{X_i}^T \boldsymbol{\beta} + \beta_0)]_+ + \lambda \sum_{j=1}^{p} |\beta_j|^2 \right\}, \tag{1.37}$$

where $\lambda > 0$ [21]. $[1 - Y_i(\boldsymbol{X_i}^T \boldsymbol{\beta} + \beta_0)]_+$ in (1.37) is called the hinge loss function. Similarly, the one-norm SVM is obtained by replacing $\|\boldsymbol{\beta}\|^2$ with $\sum_{j=1}^{p} |\beta_j|$ in (1.37):

$$\arg\min_{\boldsymbol{\beta}, \beta_0} \left\{ \sum_{i=1}^{n} [1 - Y_i(\boldsymbol{X_i}^T \boldsymbol{\beta} + \beta_0)]_+ + \lambda \sum_{j=1}^{p} |\beta_j| \right\}, \tag{1.38}$$

where $\lambda > 0$. By similar arguments, the SCAD SVM is represented as

$$\arg\min_{\boldsymbol{\beta}, \beta_0} \left\{ \sum_{i=1}^{n} [1 - Y_i(\boldsymbol{X_i}^T \boldsymbol{\beta} + \beta_0)]_+ + \sum_{j=1}^{p} p_\lambda(\beta_j) \right\}, \tag{1.39}$$

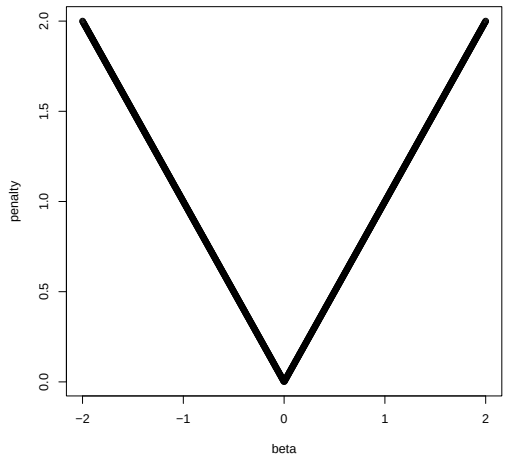

Figure 1.8: The one-norm penalty function.

where $a > 2, \lambda > 0$.

In order to solve (1.39), Zhang et al (2006) proposed a computational algorithm based on the successive quadratic algorithm for the SCAD SVM [32]. They used quadratic approximation in order to simplify the objective function to a linear equation system. They iteratively solved the linear equation system. In the studies, the SCAD SVM produced results better than the one-norm SVM in terms of prediction performance and selection accuracy.

It is known that the one-norm penalty in the one-norm SVM suffers from two limitations [30]. The first limitation is that the one-norm penalty tends to select few of them and make the rest of them 0 when predictors are highly correlated. The second limitation is that the one-norm penalty can pick at most n predictors in the $p > n$ case. It is pointed that the two-norm penalty plays a role of helping correlated predictors picked together [29]. Thus, Wang et al. (2006) considered the one-norm penalty and two-norm penalty simultaneously in the doubly regularized support vector machines below:

$$\arg\min_{\boldsymbol{\beta},\beta_0} \left\{ \sum_{i=1}^{n} [1 - Y_i(\boldsymbol{X_i}^T \boldsymbol{\beta} + \beta_0)]_+ + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \frac{\lambda_2}{2} \sum_{j=1}^{p} |\beta_j|^2 \right\}, \tag{1.40}$$

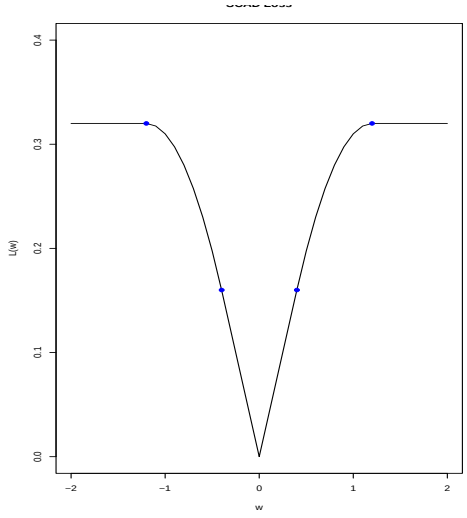

Figure 1.9: The SCAD penalty function with $\lambda = 0.4$ and $a = 3$.

where $\lambda_1 > 0, \lambda_2 > 0$ [29].

The optimization problem is the same as the following quadratic programming:

$$\min_{\beta_0, \boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \epsilon_i + \frac{\lambda_2}{2} \sum_{j=1}^{p} |\beta_j|^2 \right\}, \tag{1.41}$$

subject to

$$1 - Y_i f_i \leq \epsilon_i,$$

$$\epsilon_i \geq 0, i = 1, ..., n,$$

$$\sum_{j=1}^{p} |\beta_j| \leq \lambda_1,$$

where

$$f_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}.$$

It is noticeable that the hinge loss function and one-norm penalty are replaced by linear constraints and one-norm constraints respectively. They defined the following quadratic programming problem in order to solve the quadratic programming using Lagrange multi-

pliers and the Karush-Kuhn-Tucker (KKT) conditions:

$$\sum_{i=1}^{n}\epsilon_i+\frac{\lambda_2}{2}\sum_{j=1}^{p}|\beta_j|^2+\sum_{i=1}^{n}\alpha_i(1-Y_if_i-\epsilon_i)-\sum_{i=1}^{n}\gamma_i\epsilon_i+\eta(\sum_{j=1}^{p}|\beta_j|-\lambda_1), \tag{1.42}$$

where $\alpha_i \geq 0, \gamma_i \geq 0, \eta \geq 0$ are Lagrangian multipliers [2, 29].

Also, Wang et al. (2008) suggested the hybrid Huberized support vector machines by replacing the original hinge loss function with the Huberized hinge loss function as follows:

$$\arg\min_{\boldsymbol{\beta},\beta_0}\left\{\sum_{i=1}^{n}\phi(Y_i(\beta_0+\boldsymbol{X_i}^T\boldsymbol{\beta}))+\lambda_1\sum_{j=1}^{p}|\beta_j|+\frac{\lambda_2}{2}\sum_{j=1}^{p}|\beta_j|^2\right\}, \tag{1.43}$$

where $\lambda_1 > 0, \lambda_2 > 0$,

$$\phi(u)=\begin{cases}0, & \text{if } u>1,\\ (1-u)^2/2\delta, & \text{if } 1-\delta<u\leq 1,\\ 1-u-\delta/2, & \text{if } u\leq 1-\delta,\end{cases} \tag{1.44}$$

where $\delta \geq 0$ [30]. Most significantly, the Huberized hinge loss function is differentiable everywhere unlike the hinge loss function, which leads to significant reduction in terms of computational cost. They proposed an efficient computational algorithm to solve the entire solution path for every possible value of λ_1 with λ_2 fixed by taking an advantage of the differentiability. Most of all, the algorithm takes an advantage of the fact that the solution is a piecewise linear function with regard to λ_1.

Chapter 2

Forward Iterative Regression and Shrinkage Technique (FIRST)

The goal of the present chapter is to construct LASSO-type estimators through sequential inclusion of variables in the model with a one-dimensional absolute value penalty. Thus our procedure is essentially a forward regression method with an absolute value penalty term, which we call the *forward iterative regression and shrinkage technique* (FIRST). However, unlike LARS, our method does not give another computing algorithm for the LASSO, but actually gives a new estimator. In one dimension, our estimator agrees with the LASSO, but the two estimators are generally different in higher dimension. Though the FIRST is not an algorithm for computing the LASSO solution, it selects important variables and updates the model iteratively based on one-dimensional LASSO fittings, so it would be interesting to compare the FIRST with both LARS and the coordinate-wise descent algorithm.

2.1 Motivation

In the special case of orthogonal regressor variables, the nonnegative garrote, the LASSO and the elastic net estimator have closed-form expressions. Indeed, the non-negative

garotte with $\gamma = 1$ is given by

$$\hat{\beta}_j^G = \begin{cases} \hat{\beta}_j - \frac{\lambda}{2|\hat{\beta}_j|}, & \text{if } \hat{\beta}_j \geq \sqrt{\lambda/2}, \\ 0, & \text{if } |\hat{\beta}_j| < \sqrt{\lambda/2}, \\ \hat{\beta}_j + \frac{\lambda}{2|\hat{\beta}_j|}, & \text{if } \hat{\beta}_j \leq -\sqrt{\lambda/2}. \end{cases} \tag{2.1}$$

Similarly, for the LASSO, the corresponding expression is

$$\hat{\beta}_j^L = \begin{cases} \hat{\beta}_j - \frac{\lambda}{2}, & \text{if } \hat{\beta}_j \geq \lambda/2, \\ 0, & \text{if } |\hat{\beta}_j| < \lambda/2, \\ \hat{\beta}_j + \frac{\lambda}{2}, & \text{if } \hat{\beta}_j \leq -\lambda/2, \end{cases} \tag{2.2}$$

while for the naive elastic net, the corresponding expression is

$$\hat{\beta}_j^{EN} = \begin{cases} \frac{\hat{\beta}_j - \lambda_1/2}{1+\lambda_2}, & \text{if } \hat{\beta}_j \geq \lambda_1/2 \\ 0, & \text{if } |\hat{\beta}_j| < \lambda_1/2, \\ \frac{\hat{\beta}_j + \lambda_1/2}{1+\lambda_2}, & \text{if } \hat{\beta}_j \leq -\lambda_1/2. \end{cases} \tag{2.3}$$

In particular, since orthogonality trivially holds in one-dimension, the LASSO and the elastic net can be explicitly calculated if only one predictor is present. Therefore, if predictors are considered one at a time as in a forward selection procedure, the formulae (2.2) and (2.3) can be applied iteratively on the residual of the regression in the previous step. This leads to a new procedure, which we call the *forward iterative regression and shrinkage technique* (FIRST) and denote the resulting estimator by $\hat{\boldsymbol{\beta}}^F$.

2.2 Description

Fix a maximum number of iteration steps, say M for a fixed λ. Set initial $Y_{i,1} = Y_i$ for $i = 1, \ldots, n$ and initial prediction vector $\hat{\boldsymbol{f}}_1 = 0$. For $m = 1, \ldots, M$, repeat

1. Standardization step: Replace X_{ij} by $(X_{ij} - \bar{X}_j)/\{\sum_{i=1}^n (X_{ij} - \bar{X}_j)^2\}^{1/2}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, where $\bar{X}_j = n^{-1} \sum_{i=1}^n X_{ij}$. Thus, from now onwards, we shall assume that $\bar{X}_j = 0$ and $\sum_{i=1}^n X_{ij}^2 = 1$ for all $j = 1, \ldots, p$.

2. Regression step: Calculate ordinary least square estimates, $\hat{\beta}_{1,m}, \ldots, \hat{\beta}_{p,m}$, by $\hat{\beta}_{j,m} = \frac{\sum_{i=1}^n Y_{i,m} X_{ij}}{\sum_{i=1}^n X_{ij}^2}$, $j = 1, \ldots, p$.

3. Shrinkage step: Calculate LASSO estimates for a fixed λ,

$$\hat{\beta}^L_{j,m} = \begin{cases} \hat{\beta}_{j,m} - \lambda/2, & \text{if } \hat{\beta}_{j,m} \geq \lambda/2, \\ 0, & \text{if } |\hat{\beta}_{j,m}| < \lambda/2, \\ \hat{\beta}_{j,m} + \lambda/2, & \text{if } \hat{\beta}_{j,m} \leq -\lambda/2. \end{cases} \tag{2.4}$$

4. Selection step: Select $X_{j^*_m}$ as the most effective variable in the mth iteration step, where $j^*_m = \arg_j \min \sum_{i=1}^n (Y_{i,m} - X_{ij}\hat{\beta}^L_{j,m})^2$, $j = 1, \ldots, p$.

5. Updating stage: Update m to $m+1$, $Y_{i,m}$ to $Y_{i,m+1} = Y_{i,m} - X_{ij^*_m}\hat{\beta}^L_{j^*_m,m}$, and $\hat{\boldsymbol{f}}_m$ to $\hat{\boldsymbol{f}}_{m+1} = \hat{\boldsymbol{f}}_m + \boldsymbol{X}_{j^*_m}\hat{\beta}^L_{j^*_m,m}$.

6. Stopping rule: Stop and get out of the loop after stage m if $\|\boldsymbol{Y} - \hat{\boldsymbol{f}}_m\|^2 - \|\boldsymbol{Y} - \hat{\boldsymbol{f}}_{m+1}\|^2 < \epsilon$, where $\epsilon > 0$ is predetermined.

The final estimator is denoted by $\hat{\boldsymbol{\beta}}^F$. In practice, unless the tuned value of λ is very small, the estimator can have significant bias, and hence its predictive ability may suffer. Letting the procedure to determine the subset of predictors that are finally used in regression but determining the coefficients by an ordinary least square method using only selected predictors may significantly improve the predictive ability of the estimator while utilizing sparsity in the same way. In other words, the final estimator is given by $\hat{\boldsymbol{\beta}}^{FS} = (\boldsymbol{X}_S^T\boldsymbol{X}_S)^{-1}\boldsymbol{X}_S^T\boldsymbol{Y}$, where $S \subset \{1, \ldots, p\}$ is the subset of predictors selected by the FIRST and $X_S = (\!(X_{ij})\!)_{1 \leq i \leq n, j \in S}$. In Section 2.6, we consider the performance of $\hat{\boldsymbol{\beta}}^{FS}$ along with that of $\hat{\boldsymbol{\beta}}^F$.

In a similar manner, we may consider the forward selection version of the one-dimensional adaptive LASSO with $\gamma = 1$. The method, which will be called the adaptive FIRST, is obtained by modifying the shrinkage step in the description of the FIRST as follows:

$$\hat{\beta}^{AL}_{j,m} = \begin{cases} \hat{\beta}_{j,m} - \frac{\lambda}{2|\tilde{\beta}_j|}, & \text{if } \hat{\beta}_{j,m} \geq \frac{\lambda}{2|\tilde{\beta}_j|}, \\ 0, & \text{if } |\hat{\beta}_{j,m}| < \frac{\lambda}{2|\tilde{\beta}_j|}, \\ \hat{\beta}_{j,m} + \frac{\lambda}{2|\tilde{\beta}_j|}, & \text{if } \hat{\beta}_{j,m} \leq -\frac{\lambda}{2|\tilde{\beta}_j|}, \end{cases} \tag{2.5}$$

where $\tilde{\beta}_j$ is an initial consistent estimator of β_j. The final estimator is denoted by $\hat{\boldsymbol{\beta}}^{AF}$. Also, we can perform an ordinary least square (OLS) analysis after variable selection in

the same way as before. Then the final estimator is given by $\hat{\boldsymbol{\beta}}^{AFS} = (\boldsymbol{X}_S^T \boldsymbol{X}_S)^{-1} \boldsymbol{X}_S^T \boldsymbol{Y}$, where $S \subset \{1, \ldots, p\}$ is the subset of predictors selected by the adaptive FIRST and $X_S = (\!(X_{ij})\!)_{1 \leq i \leq n, j \in S}$.

If we incorporate an additional square of the coefficient penalty term in addition to the absolute value term in our objective function so as to minimize $\sum_{i=1}^{n}(Y_i - X_{ij}\beta_j)^2 + \lambda_1|\beta_j| + \lambda_2\beta_j^2$ with respect to β_j and $j = 1, \ldots, p$, we obtain a forward selection analog of the elastic net. The estimator, which we shall call the elastic FIRST, is obtained by modifying the shrinkage step in the algorithm for FIRST as follows:

$$\hat{\beta}_{j,m}^{EN} = \begin{cases} \frac{\hat{\beta}_{j,m} - \lambda_1/2}{1+\lambda_2}, & \text{if } \hat{\beta}_{j,m} \geq \lambda_1/2, \\ 0, & \text{if } |\hat{\beta}_{j,m}| < \lambda_1/2, \\ \frac{\hat{\beta}_{j,m} + \lambda_1/2}{1+\lambda_2}, & \text{if } \hat{\beta}_{j,m} \leq -\lambda_1/2. \end{cases} \tag{2.6}$$

The final estimator is denoted by $\hat{\boldsymbol{\beta}}^{EF}$. Naturally, one can perform an ordinary least square analysis after the variable selection stage as before. The final estimator is given by $\hat{\boldsymbol{\beta}}^{EFS} = (\boldsymbol{X}_S^T \boldsymbol{X}_S)^{-1} \boldsymbol{X}_S^T \boldsymbol{Y}$, where $S \subset \{1, \ldots, p\}$ is the subset of predictors selected by the EN and $X_S = (\!(X_{ij})\!)_{1 \leq i \leq n, j \in S}$.

2.3 Basic properties

Our method FIRST (and its adaptive and elastic variants) satisfies the following simple properties:

1. The residual sum of squares in every iteration decreases. This happens since choosing the coefficient equal to zero is equivalent to sticking with the estimate obtained in the previous stage. Since the objective criterion is minimized, one always improves the residual sum of squares using the optimizing value instead of zero.

2. The algorithm converges (even when no artificial bound on the maximum number of steps is imposed). Since zero is the definite lower bound for residual sum of squares which decreases with every iteration, the amount of decrease must be eventually small prompting algorithm to stop. Indeed, the maximum number steps are bounded above by $(\sum_{i=1}^{n} Y_i^2)/\epsilon$, where ϵ is the accuracy level chosen in the stopping rule.

3. If the regressor variables are orthogonal, no variables can be included more than one time in the selection steps. This happens because if one variable were added to the model previously, the residual variable will be uncorrelated with that and so the estimated coefficient is zero, prompting the algorithm to terminate.

4. In the orthogonal case, the maximum number of steps cannot be more than the number of variables considered for regression. Since no variables are repeated in the iterations, it is clear that there can be no more than p steps.

5. In the orthogonal case, the FIRST solution coincides with the LASSO solution and the adaptive FIRST solution coincides with the adaptive LASSO solution. In this special case, the ordinary least squares solution is $\hat{\beta}_j = \sum_{i=1}^{n} Y_i X_{ij}, j = 1, \cdots, p$. The FIRST selects the variables one by one in the order of $|\hat{\beta}_j|$'s: the larger $|\hat{\beta}_j|$, the larger the reduction in estimation error by including the corresponding variable, the earlier the corresponding variable is added to the model. If $|\hat{\beta}_j| > \frac{\lambda}{2}$, then the variable j is selected by the FIRST at some stage and its effect is estimated as $(|\hat{\beta}_j| - \frac{\lambda}{2})_+ \text{sign}(\hat{\beta}_j)$ provided the set precision level ϵ is sufficiently small. Furthermore, those variables associated with $|\hat{\beta}_j| < \frac{\lambda}{2}$ are never added to the model by the FIRST. Thus the algorithm will stop as soon as all variables j with $|\hat{\beta}_j| > \frac{\lambda}{2}$ are exhausted, and the remaining coefficients are estimated as zero. Similarly, we can show that the adaptive FIRST gives the same solution as the adaptive LASSO.

Based on the model selection result of the LASSO, the orthogonal design satisfies the irrepresentable condition [33]. Therefore the FIRST is strongly sign consistent for model selection under the orthogonal design when λ_n is chosen properly and the regularity conditions are satisfied. Also, it is known that the adaptive LASSO has the oracle properties, consistency in variable selection and asymptotic normality when the model is tuned properly [35]. Therefore we have the following theorem:

Theorem 2. *Assume the design is orthogonal with p fixed. The FIRST can select the strongly sign consistent model under the regularity conditions if* $\lambda_n/n \to 0$ *and* $\lambda_n/\sqrt{n} \to \infty$ *as* $n \to \infty$. *If the initial estimator* $\tilde{\beta}$ *is* $\sqrt{n}$ *consistent, then the* adaptive *FIRST performs as an oracle if* $\lambda_n \to \infty$ *and* $\lambda_n/\sqrt{n} \to 0$ *as* $n \to \infty$.

2.4 Recursive algorithm for the FIRST

A substantial savings in computing time of the FIRST is possible by utilizing the following recursive relations between the least square estimates $\hat{\beta}_{j,m}$ for different levels m:

$$\hat{\beta}_{j,m+1} = \hat{\beta}_{j,m} - \hat{\beta}^L_{j^*_m,m} C_{j^*_m,j}, \tag{2.7}$$

where $C_{j^*_m,j}$ is the jth element of the j^*_mth row of the correlation matrix $\boldsymbol{C}$, and the (j,k)th element of $\boldsymbol{C}$ is given by the sample correlation

$$C_{j,k} = \sum_{i=1}^{n} X_{ij} X_{ik}$$

between the jth and kth predictor. This follows by observing that $Y_{i,m+1} = Y_{i,m} - \hat{\beta}^L_{j^*_m,m} X_{ij^*_m}$ and computing inner products with X_j's. This leads to the following efficient algorithm for the computation of the FIRST. The recursive algorithm can be obtained because the ordinary least squares are iteratively calculated by the updated residuals.
For $m = 1, \ldots, M$, repeat

1. Regression step: Calculate ordinary least square estimates, $\hat{\beta}_{1,m}, \ldots, \hat{\beta}_{p,m}$, by (2.7).
2. Shrinkage step: Calculate LASSO estimates as in (2.4)
3. Selection step: Select $X_{j^*_m}$ as the most effective variable in the mth iteration step, where $j^*_m = \arg_j \min\{(\hat{\beta}^L_{j,m})^2 - 2\hat{\beta}^L_{j,m}\hat{\beta}_{j,m}\}$, $j = 1, \ldots, p$.
4. Updating stage: Update m to $m+1$ and $\hat{\boldsymbol{f}}_m$ to $\hat{\boldsymbol{f}}_{m+1} = \hat{\boldsymbol{f}}_m + \boldsymbol{X}_{j^*_m} \hat{\beta}^L_{j^*_m,m}$. Note that we have eliminated the need to compute the residuals.
5. Stopping rule: As before.

To see why the selection step 3 is the same as the selection step before, observe that

$$\begin{aligned} \sum_{i=1}^{n} (Y_{i,m} - X_{ij}\hat{\beta}^L_{j,m})^2 &= \sum_{i=1}^{n} Y^2_{i,m} - 2\hat{\beta}^L_{j,m} \sum_{i=1}^{n} X_{ij} Y_{i,m} + (\hat{\beta}^L_{j,m})^2 \sum_{i=1}^{n} X^2_{ij} \\ &= \sum_{i=1}^{n} Y^2_{i,m} - 2\hat{\beta}^L_{j,m}\hat{\beta}_{j,m} + (\hat{\beta}^L_{j,m})^2. \end{aligned}$$

The assertion follows since the first term does not depend on j. In a similar manner, efficient algorithms for the *adaptive* FIRST (respectively, the *elastic* FIRST) are obtained by replacing $\hat{\beta}^{L}_{j^*_m,m}$ in (2.7) by $\hat{\beta}^{AL}_{j^*_m,m}$ (respectively, $\hat{\beta}^{EN}_{j^*_m,m}$) and replacing (2.4) in the shrinkage step by (2.5) (respectively, (2.6)).

2.5 Relations with $L2$Boosting

Boosting is another iterative approach for building sparse models for high dimensional data. Friedman (1999) gave a nice interpretation of boosting as a gradient tree boosting for multiple additive regression trees [16]. In the gradient tree boosting, the regression trees are iteratively fitted to the steepest descent gradients giving terminal regions in the gradient tree boosting. Bühlmann (2006) considered boosting with the squared error loss, called L_2Boosting, and showed that L_2Boosting for linear models produces consistent prediction estimates [9]. The L_2Boosting is a very attractive procedure in the LPSN context, since it just solves a series of simple least squares regression problems. Bühlmann and Hothorn (2007) pointed out differences between L_2Boosting and the LASSO [10]. When we add some backward steps to L_2Boosting, the modified L_2Boosting coincides with the LASSO [34]. Both boosting and the FIRST are algorithm-based methods which build the model iteratively, however, they work differently in several ways.

Firstly, the L_2Boosting repeatedly fits ordinary least squares for residuals to update the model, while the FIRST uses the LASSO fittings to update the model. Secondly, these two procedures use different schemes to update the model between iterations. At each step, the L_2Boosting updates the model by adding the new term multiplying a small step size, say $\nu = 0.1$, which works as a shrinkage parameter to scale down the contribution of the newly added term. By contrast, the FIRST added a new term with the constant step size one, since the new term is already subject to a soft-thresholding penalty via the LASSO before being added to the model. Thirdly, the FIRST has a stopping rule whereas L_2Boosting does not have any stopping rule. More precisely, the number of boosting iterations is an important tuning parameter for the L_2Boosting, which directly determines the size of the final model and needs to be selected properly to avoid overfitting [9, 10, 11, 12]. In the FIRST, λ is the tuning parameter which automatically incorporates the

amount of shrinkage at each step and hence controls the goodness of fit for the final model. Consequently, λ also controls the stopping rule.

2.6 Simulation

In this section, we demonstrate the performance of the FIRST method and its variations in different settings. Since the FIRST method is designed for large p small n problems, we simulate the data from a high-dimensional sparse regression model

$$Y_i = \sum_{j=1}^{n} \boldsymbol{X}_i^T \beta + \sigma\varepsilon_i, \quad i = 1, ..., n, \quad \varepsilon_i \sim \text{i.i.d. } N(0,1).$$

where $\boldsymbol{X}_i \in \mathbb{R}^p$ and $p > n$. The simulated data consist of a training set, a validation set, and an independent test set of size 1,000. We assume the training and tuning sets have the same sample size n. For each method, we fit the model using the training data and use the validation set to select the tuning parameter. The mean squared error on the test set,

$$\frac{\sum_{i=1}^{1000}(Y_i - \hat{Y}_i)^2}{1000}$$

is used to evaluate prediction performance of each method. We run 100 simulations for each experiment and report the average results.

We implement six variations of the FIRST method, including the FIRST, the adaptive FIRST (aFIRST), the elastic FIRST (eFIRST), FIRST followed by the OLS (FIRST+OLS), the adaptive FIRST followed by the OLS (aFIRST+OLS), and the elastic FIRST followed by the OLS (eFIRST+OLS). For comparison, we also include the LASSO and elastic net (EN) results. We implement our algorithms in R. Two algorithms are used to implement the LASSO and EN: the LARS and the coordinate-wise descent algorithm, both available in R. The optimal tuning parameter for each method is chosen by a grid search using the validation set. All simulations are run on Dell Xeon Dual Core 3.6 GHz with 4096 MB RAM.

2.6.1 Simulation settings

We present five examples, which consider different sample sizes, error variances (or signal strength), and correlation scenarios among $\boldsymbol{X}$. Here are the details of four examples.

(a) In Example 1, we have $p = 1000$ and $n = 100,500$. All the covariates $X_1, \ldots, X_p$ are i.i.d from $N(0,1)$. The variance for the error $\sigma^2 = 1$. The true coefficient vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ contains 10 non-zero coefficients, (3, 3, 3, 3, 1.5, 1.5, 1.5, 2, 2, 2), and the rest are zero coefficients. We let the locations of the non-zero coefficients be equally spaced. In other words, β_1, β_{101}, β_{201} and β_{301} are 3, β_{401}, β_{501} and β_{601} are 1.5, and β_{701}, β_{801} and β_{901} are 2. $\epsilon = 0.1$ and $M = 20$ are used.

(b) Example 2 is the same as Example 1, except that there is moderate correlation among important covariates. In particular, the pairwise correlation between any two important covariates X_i and X_j is $\text{corr}(X_i, X_j) = \rho^{|i-j|/100}$ where $\rho = 0.5$ if $i, j = 1, 101, 201, ..., 901$. The values $\epsilon = 0.001$ and $M = 50$ are used.

(c) Example 3 had the same setting as Example 1 only with $n = 100$, except that all the covariates are moderately correlated: the pairwise correlation between X_i and X_j is $\text{corr}(X_i, X_j) = \rho^{|i-j|}$ for any $i, j = 1, 2, ..., 1000$. Here $\rho = 0.5$. We consider two different variances: $\sigma^2 = 1$ and $\sigma^2 = 4$. The values $\epsilon = 0.001$ and $M = 50$ are used.

(d) Example 4 is the same as Example 3, except that the covariates are highly correlated with $\rho = 0.9$. The values $\epsilon = 0.001$ and $M = 50$ are used.

(e) Example 5 is the same as Example 1, except that $p = 2000$. The values $\epsilon = 0.1$ and $M = 20$ are used.

2.6.2 Experiments and results

We compare ten different methods with regard to their prediction error, variable selection performance, and computation time. In particular, the mean squared error on the test set is used to evaluate their prediction performance. For variable selection, we report two types of selection errors. Selection error I is defined as the number of non-zero coefficients which are estimated as zero, and selection error II is the number of zero coefficients which are not estimated as zero. The computation cost is reported as the average time (in seconds) to obtain the final coefficients in one run.

Tables 2.1 and 2.2 summarize the result for the simple independent case, while Tables 2.3 and 2.4 assume there is moderate correlation among important covariates. It

is observed that the FIRST algorithms give quite competitive performance compared with the LASSO, when p is larger than n. In the first example, the adaptive FIRST followed by the OLS works best in terms of the test error whereas the elastic FIRST followed by the OLS does in the second example. One may wonder why the FIRST followed by the OLS has smaller selection error when the OLS simply re-estimates the regression coefficients after selection by the FIRST. We think the reasons are that these two procedures select tuning parameters differently as well as stop at different times. Since a follow-up OLS step generally improves quality of estimates of regression coefficients by reducing bias significantly, the FIRST followed by the OLS has a better prediction power prompting it to stop sooner without including some of the unimportant variables otherwise selected by the FIRST. A similar explanation applies to the comparison between the adaptive FIRST and the adaptive FIRST followed by the OLS. Interestingly, the LASSO solution obtained by the CDA and that by the LARS give very similar test errors, but the selection performance of the LASSO by the CDA is much worse. It is noted that the LASSO by the CDA tends to retain a large number of redundant variables in the final model. The standard EN results, implemented by either the LARS or the CDA, are not so good as other methods for independent cases, but they work better for correlated cases as shown in Table 2.3 and 2.4. One explanation is that the EN methods are specially designed for handling correlated covariates.

With regard to the computation cost, the LASSO by the CDA is the best. One may note another anomaly that the computing time for the adaptive FIRST and the adaptive FIRST followed by the OLS in Tables 2.1–2.4 actually decreased when the sample size increased from $n = 100$ to $n = 500$. It may be noted that the role of n in computing time is limited only to the standardization step and the first step, while the dimension p plays a far more important role. Since in our examples n is much smaller than p, there may not be any significant impact of a larger value of n on computing time compared to computational burden in subsequent steps, in each of which $O(p)$ calculations are needed. Moreover, as our iterative procedures are dependent on stopping times and relatively sooner convergence is expected for a more accurate procedure associated with higher sample size, it is possible to observe shorter computing time for larger sample size. Finally, computing time can vary up to some extent randomly depending on number of jobs running on the server where all our programs ran. We also notice that, as n increases from 100 to 500, the computation

Table 2.1: Simulation results for Example 1, $n = 100$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	2.46(0.57)	0	6.24	13.95
aFIRST	2.31(0.60)	0	5.87	12.36
eFIRST	2.30(0.58)	0	5.28	72.20
FIRST+OLS	1.38(0.28)	0	3.20	13.95
aFIRST+OLS	1.20(0.15)	0	0.32	12.36
eFIRST+OLS	1.42(0.22)	0	1.44	72.20
LASSO (LARS)	2.59(0.83)	0	55.03	1.47
EN (LARS)	3.28(0.90)	0	63.54	428.80
LASSO (CDA)	2.57(0.81)	0	92.01	0.36
EN (CDA)	2.92(0.95)	0	129.1	2.70

time of the LASSO and EN given by the LARS algorithm seriously deteriorates, while other algorithms are not so significantly affected by the sample size. Tables 2.5–2.8 respectively consider the scenarios where all the covariates are either moderately or highly correlated. We notice that the aFIRST+OLS works best when the covariates are moderately correlated, while the eFIRST+OLS is the best in the high correlation cases. The LASSO and EN by the CDA are the fastest, but selection error II is the worst among all. The EN by the LARS is the slowest procedure. Overally speaking, the FIRST algorithms lead to much leaner models with an improved prediction performance than the LASSO and EN solutions. Additionally, we considered $p = 2000$ in Example 5. The results are shown in Tables 2.9–2.10. As expected by us, the performance of all the methods get better when the sample size increases and get worse when the number of redundant predictors increases.

Table 2.2: Simulation results for Example 1, $n = 500$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	1.17(0.06)	0	5.17	14.60
aFIRST	1.15(0.07)	0	3.99	11.23
eFIRST	1.16(0.07)	0	5.37	75.70
FIRST+OLS	1.02(0.04)	0	0.09	14.60
aFIRST+OLS	1.02(0.04)	0	0.04	11.23
eFIRST+OLS	1.07(0.05)	0	0.01	75.70
LASSO (LARS)	1.15(0.05)	0	41.01	48.37
EN (LARS)	1.76(0.68)	0	24.31	755.60
LASSO (CDA)	1.16(0.05)	0	86.07	1.08
EN (CDA)	1.20(0.08)	0	67.53	6.00

Table 2.3: Simulation results for Example 2, $n = 100$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	13.47(6.88)	0.86	28.61	414.2
aFIRST	13.34(6.58)	1.54	15.24	401.8
eFIRST	2.80(0.82)	0	14.41	417.0
FIRST+OLS	4.24(2.74)	0.96	8.16	414.2
aFIRST+OLS	6.14(3.81)	0	6.15	401.9
eFIRST+OLS	1.40(0.25)	0.01	0.58	417.0
LASSO (LARS)	1.82(0.40)	0	26.78	1.7
EN (LARS)	1.95(0.45)	0	24.14	454.9
LASSO (CDA)	1.82(0.40)	0	26.92	0.6
EN (CDA)	1.82(0.41)	0	36.02	3.7

Table 2.4: Simulation results for Example 2, $n = 500$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	1.39(0.11)	0	13.43	423.3
aFIRST	1.37(0.10)	0	14.75	401.2
eFIRST	1.32(0.10)	0	10.83	426.3
FIRST+OLS	1.08(0.07)	0	0.05	423.4
aFIRST+OLS	1.08(0.07)	0	0.02	401.2
eFIRST+OLS	1.08(0.07)	0	0.17	426.3
LASSO (LARS)	1.14(0.08)	0	32.99	47.7
EN (LARS)	1.45(0.49)	0	195.12	739.6
LASSO (CDA)	1.14(0.08)	0	21.70	1.4
EN (CDA)	1.14(0.08)	0	28.36	7.4

Table 2.5: Simulation results for Example 3, $\sigma^2 = 1$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	2.85(1.17)	0	32.69	460.7
aFIRST	2.48(1.12)	0	14.17	440.1
eFIRST	2.67(1.12)	0	29.35	464.7
FIRST+OLS	1.73(0.67)	0	6.82	460.7
aFIRST+OLS	1.40(0.24)	0	0.49	440.1
eFIRST+OLS	1.61(0.54)	0.01	4.97	464.7
LASSO (LARS)	2.95(1.12)	0	52.80	2.6
EN (LARS)	3.03(1.17)	0	49.87	624.2
LASSO (CDA)	2.96(1.14)	0	79.86	0.9
EN (CDA)	2.96(1.14)	0	79.86	5.4

Table 2.6: Simulation results for Example 3, $\sigma^2 = 4$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	9.85(4.43)	0.09	34.11	477.4
aFIRST	8.55(4.45)	0.14	12.49	462.2
eFIRST	9.20(3.85)	0.05	32.74	481.1
FIRST+OLS	7.94(3.28)	0.22	14.34	477.4
aFIRST+OLS	5.94(2.34)	0	1.87	462.2
eFIRST+OLS	7.20(2.61)	0.14	12.35	481.1
LASSO (LARS)	10.81(3.68)	0.08	51.28	2.8
EN (LARS)	10.84(3.71)	0.07	52.21	630.1
LASSO (CDA)	10.84(3.71)	0.08	60.80	1.0
EN (CDA)	10.84(3.71)	0.08	60.80	5.5

Table 2.7: Simulation results for Example 4, $\sigma^2 = 1$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	7.97(5.15)	1.76	31.30	464.0
aFIRST	8.18(5.47)	2.36	18.33	448.8
eFIRST	5.47(3.08)	0.55	27.38	470.1
FIRST+OLS	4.08(2.87)	1.56	13.56	464.0
aFIRST+OLS	4.73(3.76)	0	5.84	448.8
eFIRST+OLS	2.07(1.02)	0.38	13.36	470.1
LASSO (LARS)	2.60(0.73)	0	49.14	3.8
EN (LARS)	2.66(0.75)	0	50.08	625.4
LASSO (CDA)	2.61(0.73)	0	50.24	1.0
EN (CDA)	2.61(0.73)	0	50.24	5.8

Table 2.8: Simulation results for Example 4, $\sigma^2 = 4$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	13.53(4.98)	2.58	31.84	439.1
aFIRST	13.18(5.38)	3.07	14.98	424.1
eFIRST	11.42(3.49)	1.47	31.05	444.0
FIRST+OLS	10.51(3.94)	2.74	15.26	439.1
aFIRST+OLS	9.99(3.95)	0	7.08	424.1
eFIRST+OLS	7.83(2.30)	1.49	15.09	440.0
LASSO (LARS)	9.27(2.01)	0.54	48.88	3.3
EN (LARS)	9.32(2.07)	0.56	48.06	557.8
LASSO (CDA)	9.28(2.00)	0.54	49.79	0.9
EN (CDA)	9.29(2.01)	0.54	50.19	5.9

Table 2.9: Simulation results for Example 5, $n = 100$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	3.07(2.09)	0.01	6.96	28.6
aFIRST	2.86(2.18)	0.01	6.17	25.6
eFIRST	2.73(1.07)	0	5.91	147.81
FIRST+OLS	1.64(0.68)	0.01	4.64	28.6
aFIRST+OLS	1.27(0.48)	0	0.48	25.6
eFIRST+OLS	1.53(0.28)	0	2.84	147.81
LASSO (LARS)	3.53(1.52)	0.01	60.65	2.63
EN (LARS)	5.27(2.60)	0.03	49.57	3074.4
LASSO (CDA)	3.54(1.53)	0.01	274.04	0.9
EN (CDA)	4.19(2.71)	0.02	348.32	668.4

Table 2.10: Simulation results for Example 5, $n = 500$

Method	Test Error	Selection Error I	Selection Error II	Time(sec.)
FIRST	1.19(0.09)	0	5.72	28.9
aFIRST	1.15(0.09)	0	5.44	22.4
eFIRST	1.18(0.07)	0	5.60	147.62
FIRST+OLS	1.03(0.05)	0	0	28.9
aFIRST+OLS	1.03(0.05)	0	0	22.4
eFIRST+OLS	1.07(0.05)	0	0.01	147.62
LASSO (LARS)	1.16(0.07)	0	63	75.8
EN (LARS)	2.00(0.57)	0	373.02	4044
LASSO (CDA)	1.20(0.07)	0	84.27	3.5
EN (CDA)	1.24(0.08)	0	84.98	1332.6

2.7 Real data example

We consider the gene expression data and approaches used in Huang et al. (2008) [23]. There are totally 31,099 probe sets and 120 observations in this dataset. For high dimensional data like this, it is a common practice to use pre-screening to make the computation more manageable. Two stages of pre-screening were applied in our analysis. In the first pre-screening, we removed 3,815 probe sets whose maximum expression values are not greater than the 25th percentile of the entire probe sets. In the second stage, we selected 3,000 probe sets with the largest variances among the remaining 27,283 probe sets. Then in our analysis, we used these 3,000 probe sets the predictors. Since these 3,000 predictors are very likely to be correlated with each other, we implemented the elastic FIRST and the elastic FIRST followed by the OLS, with $\epsilon = 0.001,$ and $M = 200$. The elastics net (EN) was also implemented respectively by the LARS and the coordinate-wise descent algorithm (CDA). We randomly select 100 training data and 20 test data. Five-fold cross validation is conducted with 100 training data in R. The test error, the number of non-zero estimates, and computation time are shown in Table 2.11. We observe that the elastic FIRST (eFIRST) gives the smallest test error, the EN by the CDA is the second best, and the EN by the LARS is the worst. In terms of the model size, the eFIRST followed by the OLS gives the most sparse model of size 7, the eFIRST and the EN produce similar model sizes 36 and 31, while the EN by the CDA gives the largest model of size $2,095$. With regard to

Table 2.11: Real example results

Method	Test Error	Selected Genes	Time(min.)
eFIRST	0.00828	36	52.94
eFIRST+OLS	0.01057	7	52.94
EN (LARS)	0.01216	31	1098.32
EN (CDA)	0.00866	2095	3.22

the computation time, the EN by the LARS struggles with the problem and take almost 20 times longer of that of the elastic FIRST. On the other hand, the EN by the CDA is the fastest in terms of computing time.

2.8 Discussion

We propose a new variable selection algorithm for high dimensional sparse regression models and the recursive algorithm for computational reduction. Basically, the FIRST is a combination of one dimensional LASSO and forward selection. The FIRST takes an advantage of the closed-form solutions for the one-dimensional LASSO and forward selection of fitting residuals repeatedly. Furthermore, we extend the FIRST to the adaptive FIRST and the elastic FIRST by applying the same concepts to the adaptive LASSO and the EN. We also consider an ordinary least square operation after applying the FIRST, the adaptive FIRST and the elastic FIRST. We finally derive a recursive algorithm from the relations between the successive least square estimates and residuals, which leads to substantial savings in computing time. Throughout the simulation study and a real data example, we show that our algorithms generally show good prediction performance and have selection accuracy in comparison with the LASSO and the EN. Our algorithms also have reasonable computation cost, and we expect that they can be utilized for real high dimensional sparse data especially. As a future work, we will make an effort to tune parameters more efficiently for more computational savings.

Chapter 3

Forward Iterative Selection and Classification Algorithm (FISCAL)

The goal of the present chapter is to construct SVM-type estimators through sequential inclusion of variables in the model with a one-dimensional squared hinge loss function. Thus our procedure is essentially a forward classification method with a squared hinge loss function, which we call the *forward iterative selection and classification algorithm* (FISCAL). We suggest the squared SVM by squaring the original hinge loss function and using ℓ_1-norm and ℓ_2-norm simultaneously for the FISCAL. The FISCAL selects important variables and updates the model iteratively based on one-dimensional squared SVM fittings. Additionally, we consider the FISCAL using one-dimensional one-norm SVM.

3.1 Motivation

We consider a variation of the SVM by the penalty terms of combining (1.37) and (1.38) and modify the misclassification error to its square where we can remove β_0 by centering. The reason for considering the square is to make the error term differentiable in $\boldsymbol{\beta}$ so that minimizers may be found by calculus. Then we can consider both one-norm penalty and two-norm penalty at the same time. It is known that one-norm penalty in the one-norm SVM is good for feature suppression and two-norm penalty in the original SVM plays a role of helping correlated predictors picked together. So we can expect an enhanced SVM by combining both penalty terms:

$$\arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} [1 - Y_i \boldsymbol{X}_i^T \boldsymbol{\beta}]_+^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} {\beta_j}^2 \right\}, \tag{3.1}$$

where $\lambda_1 > 0, \lambda_2 > 0$. We call this combined SVM the squared SVM. If there is only one predictor, X_j in (3.1), the problem reduces to finding

$$\arg\min_{\beta_j} \left\{ \sum_{i=1}^{n} [1 - Y_i X_{ij} \beta_j]_+^2 + \lambda_1 |\beta_j| + \lambda_2 {\beta_j}^2 \right\}, \tag{3.2}$$

where $\lambda_1 > 0, \lambda_2 > 0$. Since (3.2) has a solution in one-dimension which can be described relatively easily, the squared SVM can be explicitly solved if only one predictor is present. Therefore, if predictors are considered one at a time as in a forward selection procedure, (3.2) can be applied iteratively in the previous step after taking into account of the variables present from the earlier steps. This leads to a new procedure, which we call the *forward iterative selection and classification algorithm* (FISCAL) and denote the resulting estimator by $\hat{\boldsymbol{\beta}}^F$.

3.2 Description

Fix a maximum number of iteration steps, say M for a fixed λ_1 and λ_2. Set initial $c_{i,1} = 1$ for $i = 1, \ldots, n$ and initial coefficient vector $\hat{\boldsymbol{\beta}}_{\mathbf{1}} = 0$.
For $m = 1, \ldots, M$, repeat

1. Standardization step: Replace X_{ij} by $(X_{ij} - \bar{X}_j)/\{\sum_{i=1}^{n}(X_{ij} - \bar{X}_j)^2\}^{1/2}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, where $\bar{X}_j = n^{-1} \sum_{i=1}^{n} X_{ij}$. Thus, from now onwards, we shall assume that $\bar{X}_j = 0$ and $\sum_{i=1}^{n} X_{ij}^2 = 1$ for all $j = 1, \ldots, p$.

2. SVM step: Solve the following squared SVM for $\hat{\beta}_{1,m}, \ldots, \hat{\beta}_{p,m}$, $j = 1, \ldots, p$, after standardization :

$$\hat{\beta}_{j,m} \;=\; \arg\min_{b_{j,m}} \left\{ \sum_{i=1}^{n} [c_{i,m} - Y_i X_{ij} b_{j,m}]_+^2 + \lambda_1 |b_{j,m}| + \lambda_2 b_{j,m}^2 \right\}, \tag{3.3}$$

   where $\lambda_1 > 0, \lambda_2 > 0$ are given tuning parameters.

3. Selection step: Select $X_{j_m^*}$ as the most effective variable in the mth iteration step where $j_m^* = \arg_j \min\{\sum_{i=1}^{n} [c_{i,m} - Y_i X_{ij} \hat{\beta}_{j,m}]_+^2\}$, $j = 1, \ldots, p$.

4. Updating stage: Update m to $m+1$, $c_{i,m}$ to $c_{i,m+1} = c_{i,m} - Y_i X_{ij^*_m} \hat{\beta}_{j^*_m,m}$, and $\hat{\boldsymbol{\beta}}_{\boldsymbol{m}}$ to $\hat{\boldsymbol{\beta}}_{\boldsymbol{m+1}} = \hat{\boldsymbol{\beta}}_{\boldsymbol{m}} + (0, ...\hat{\beta}_{j^*_m,m}..., 0)^T$.

5. Stopping rule: Stop after stage m and get out of the loop if $\{\sum_{i=1}^n [c_{i,m}]_+^2\} - \{\sum_{i=1}^n [c_{i,m+1}]_+^2\} < \epsilon$, where $\epsilon > 0$ is predetermined.

The final estimator is denoted by $\hat{\boldsymbol{\beta}}^F$. The special case corresponding to $\lambda_2 = 0$ is of particular interest. This choice makes the task of choosing the tuning parameter easier since now only one tuning parameter λ_1 needs to be chosen, while still retaining the ability to select variables. Of course, better prediction power may be achieved by a full tuning with both λ_1 and λ_2.

3.3 Optimization in the squared SVM step

The objective function in the squared SVM in the Section 3.2 is

$$F(b_j) = \sum_{i=1}^{n} [c_i - Y_i X_{ij} b_j]_+^2 + \lambda_1 |b_j| + \lambda_2 b_j^2. \tag{3.4}$$

This objective function is a quadratic and convex function in b_j, and is differentiable except at zero. The derivative of the objective function is given by

$$F'(b_j) = -2 \sum_{i=1}^{n} [c_i - Y_i X_{ij} b_j]_+ Y_i X_{ij} + \lambda_1 \text{sign}(b_j) + 2\lambda_2 b_j, \text{ if } b_j \neq 0 \tag{3.5}$$

The first term in (3.5) can be simplified by separating into cases assuming that $Y_1, ..., Y_N = +1, Y_{N+1}, ..., Y_n = -1$ for simplification and computational savings:

1. For $(X_{ij} > 0)$

$$-2 \sum_{i=1}^{N} (c_i X_{ij} - X_{ij}^2 b_j) I(c_i X_{ij}^{-1} > b_j, X_{ij} > 0) \tag{3.6}$$
$$+2 \sum_{i=N+1}^{n} (c_i X_{ij} + X_{ij}^2 b_j) I(c_i X_{ij}^{-1} > -b_j, X_{ij} > 0);$$

2. For $(X_{ij} < 0)$

$$-2 \sum_{i=1}^{N} (c_i X_{ij} - X_{ij}^2 b_j) I(c_i X_{ij}^{-1} < b_j, X_{ij} < 0) \tag{3.7}$$
$$+2 \sum_{i=N+1}^{n} (c_i X_{ij} + X_{ij}^2 b_j) I(c_i X_{ij}^{-1} < -b_j, X_{ij} < 0);$$

where $i = 1, ..., N$, are observations for class $+1$ and $i = N+1, ..., n$ are observations for class -1 after a possible rearrangement of observations.

The derivative of the objective function is an increasing function which is continuous except at zero. In particular, the derivative function passes through zero from the negative side to the positive side only once. In addition, since the derivative function is dependent on the indicator functions, the above functional forms can change only at $b_j = 0$ or $b_j = \frac{c_i}{X_{ij}}$ where $i = 1, ..., N$, are observations for class $+1$ or $b_j = \frac{-c_i}{X_{ij}}$ where $i = N+1, ..., n$, are observations for class -1. This helps a lot to locate the minimum, since the derivative is linear in between these points. Let $B_{-k_2}, ..., B_{-1}, B_1, ..., B_{k_1}$ be distinct ordered values of $\frac{c_i}{X_{ij}}$ and $\frac{-c_i}{X_{ij}}$ where $B_{-k_2} < ... < B_{-1} < 0 < B_1 < ... < B_{k_1}$. Then we can locate the optimal point by the algorithm given below:

1. Calculate
$$F'(0+) = -2\sum_{i=1}^{n}\{\max(c_i, 0)Y_iX_{ij}\} + \lambda_1,$$
$$F'(0-) = -2\sum_{i=1}^{n}\{\max(c_i, 0)Y_iX_{ij}\} - \lambda_1.$$

2. If $F'(0-) < 0 < F'(0+)$, set $\hat{\beta}_{j,m} = 0$ and stop.

3. Else if $F'(0-) > 0$, observe $F'(B_{-1}), ..., F'(B_{-l})$ until $F'(B_{-l}) < 0$ and set
$$\hat{\beta}_{j,m} = B_{-l} - F'(B_{-l})\frac{B_{-(l-1)} - B_{-l}}{F'(B_{-(l-1)}) - F'(B_{-l})} \ . \tag{3.8}$$

4. Else if $F'(0+) < 0$, observe $F'(B_1), ..., F'(B_l)$ until $F'(B_l) > 0$ and set
$$\hat{\beta}_{j,m} = B_{l-1} - F'(B_{l-1})\frac{B_l - B_{l-1}}{F'(B_l) - F'(B_{l-1})} \ . \tag{3.9}$$

Figures 3.1 - 3.3 graphically explain how to locate the optimal point in the three cases, $F'(0-) < 0 < F'(0+)$, $F'(0-) > 0$ and $F'(0+) < 0$ in the algorithm. In Figure 3.4, we can calculate the optimal point, ξ by using the following equation:

$$\xi = \xi_1 - \frac{C_1}{C_2 - C_1}(\xi_2 - \xi_1) \tag{3.10}$$

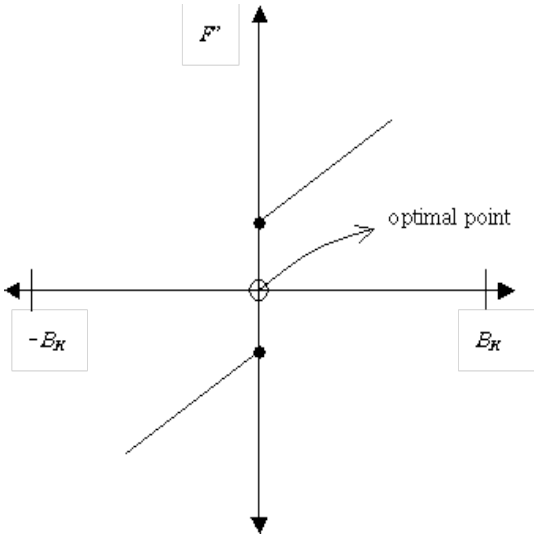

Figure 3.1: Case 1 : $F'(0-) < 0 < F'(0+)$

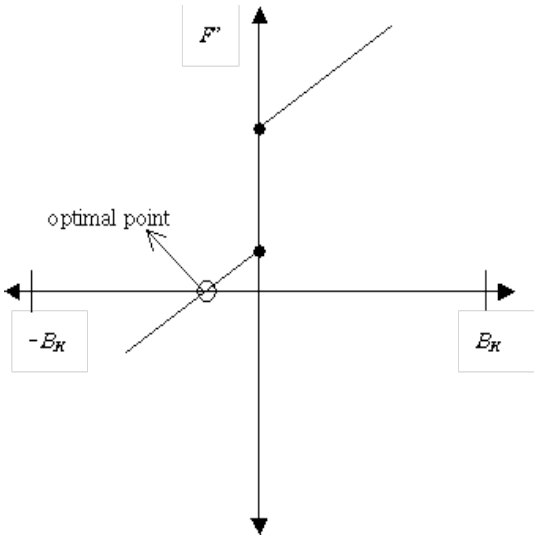

Figure 3.2: Case 2 : $F'(0-) > 0$

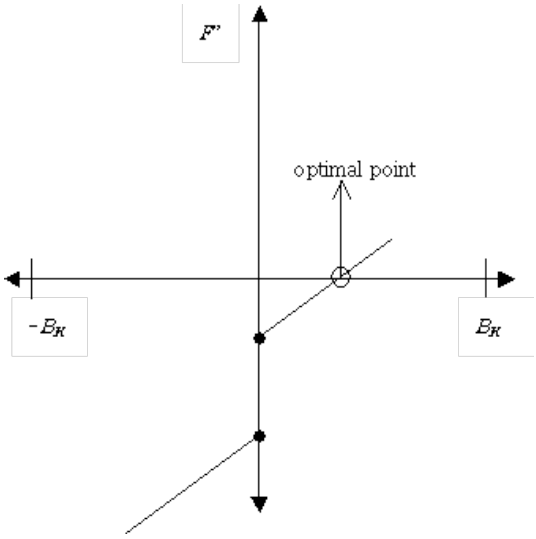

Figure 3.3: Case 3 : $F'(0+) < 0$

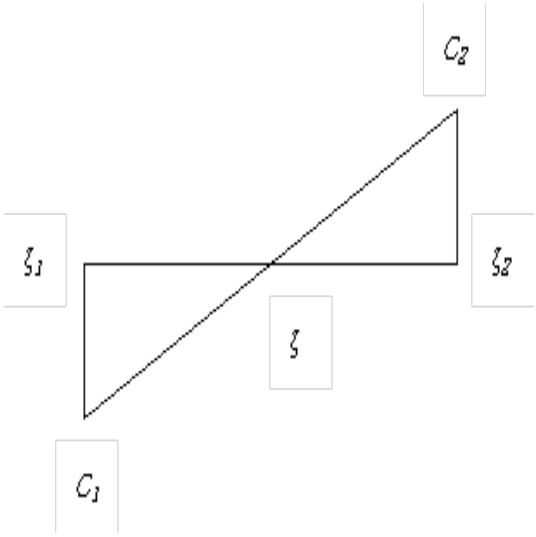

Figure 3.4: Interpolation

By using (3.10), we can interpolate the optimal point in the two cases, $F'(0-) > 0$ and $F'(0+) < 0$ in the algorithm.

3.4 Recursive algorithm for the squared SVM step

A substantial savings in computing time of the FISCAL is possible by utilizing a recursive relation to relating the sums in (3.6) - (3.7).

Observe that

1. For $b_j > 0$,

$$F'(b_j) = -2T_1 + 2b_jT_2 + 2T_3 + 2b_jT_4 - 2T_5 + 2b_jT_6 + 2T_7 + 2b_jT_8 + \lambda_1 + 2\lambda_2 b_j, \quad (3.11)$$

where

$$T_1 = \sum_{i=1}^{N} c_i X_{ij} I(c_i X_{ij}^{-1} > b_j, b_j > 0, X_{ij} > 0), \quad (3.12)$$

$$T_2 = \sum_{i=1}^{N} X_{ij}^2 I(c_i X_{ij}^{-1} > b_j, b_j > 0, X_{ij} > 0), \quad (3.13)$$

$$T_3 = \sum_{i=N+1}^{n} c_i X_{ij} I(c_i X_{ij}^{-1} > -b_j, b_j > 0, X_{ij} > 0), \quad (3.14)$$

$$T_4 = \sum_{i=N+1}^{n} X_{ij}^2 I(c_i X_{ij}^{-1} > -b_j, b_j > 0, X_{ij} > 0), \quad (3.15)$$

$$T_5 = \sum_{i=1}^{N} c_i X_{ij} I(c_i X_{ij}^{-1} < b_j, b_j > 0, X_{ij} < 0), \quad (3.16)$$

$$T_6 = \sum_{i=1}^{N} X_{ij}^2 I(c_i X_{ij}^{-1} < b_j, b_j > 0, X_{ij} < 0), \quad (3.17)$$

$$T_7 = \sum_{i=N+1}^{n} c_i X_{ij} I(c_i X_{ij}^{-1} < -b_j, b_j > 0, X_{ij} < 0), \tag{3.18}$$

$$T_8 = \sum_{i=N+1}^{n} X_{ij}^2 I(c_i X_{ij}^{-1} < -b_j, b_j > 0, X_{ij} < 0). \tag{3.19}$$

2. For $b_j < 0$,

$$F'(b_j) = -2T_1 + 2b_j T_2 + 2T_3 + 2b_j T_4 - 2T_5 + 2b_j T_6 + 2T_7 + 2b_j T_8 - \lambda_1 + 2\lambda_2 b_j, \tag{3.20}$$

where

$$T_1 = \sum_{i=1}^{N} c_i X_{ij} I(c_i X_{ij}^{-1} > b_j, b_j < 0, X_{ij} > 0), \tag{3.21}$$

$$T_2 = \sum_{i=1}^{N} X_{ij}^2 I(c_i X_{ij}^{-1} > b_j, b_j < 0, X_{ij} > 0), \tag{3.22}$$

$$T_3 = \sum_{i=N+1}^{n} c_i X_{ij} I(c_i X_{ij}^{-1} > -b_j, b_j < 0, X_{ij} > 0), \tag{3.23}$$

$$T_4 = \sum_{i=N+1}^{n} X_{ij}^2 I(c_i X_{ij}^{-1} > -b_j, b_j < 0, X_{ij} > 0), \tag{3.24}$$

$$T_5 = \sum_{i=1}^{N} c_i X_{ij} I(c_i X_{ij}^{-1} < b_j, b_j < 0, X_{ij} < 0), \tag{3.25}$$

$$T_6 = \sum_{i=1}^{N} X_{ij}^2 I(c_i X_{ij}^{-1} < b_j, b_j < 0, X_{ij} < 0), \tag{3.26}$$

$$T_7 = \sum_{i=N+1}^{n} c_i X_{ij} I(c_i X_{ij}^{-1} < -b_j, b_j < 0, X_{ij} < 0), \tag{3.27}$$

$$T_8 = \sum_{i=N+1}^{n} X_{ij}^2 I(c_i X_{ij}^{-1} < -b_j, b_j < 0, X_{ij} < 0). \tag{3.28}$$

The recursive algorithm can be derived based on the natural inclusion-exclusion relations in the following 8 sets between the two consecutive knots where $B_{-k_2} < ... < B_{-1} < 0 < B_1 < ... < B_{k_1}$.

First, we define 4 sets in order to derive the following recursive relations:

1. SET 1: $I_1 = \{i \ : \ c_i X_{ij}^{-1} = B_{K_1}\}$.
2. SET 2: $I_2 = \{i \ : \ c_i X_{ij}^{-1} = B_{K_1-1}\}$.
3. SET 3: $I_3 = \{i \ : \ c_i X_{ij}^{-1} = -B_{K_1}\}$.
4. SET 4: $I_4 = \{i \ : \ c_i X_{ij}^{-1} = -B_{K_1-1}\}$.

Consider $F'(\cdot)$ on the positive side where $b_j > 0$. Let $K_1 = 1, ..., k_1$. For instance, when b_j changes from B_{K_1-1} to B_{K_1}, $K_1 = 1, ..., k_1$, the index i drops from the set in (3.12) and T_1 decreases by $c_i X_{ij}$ since the inequality, $c_i X_{ij}^{-1} > b_j$ is now violated. Other sets are affected similarly leading to the following relations:

$$\text{If } I_1 \neq \emptyset, T_1 \text{ is replaced by } T_1 - \sum_{i \in I_1} c_i X_{ij}, \text{ else } T_1 \text{ is unchanged,} \tag{3.29}$$

$$\text{If } I_1 \neq \emptyset, T_2 \text{ is replaced by } T_2 - \sum_{i \in I_1} X_{ij}^2, \text{ else } T_2 \text{ is unchanged,} \tag{3.30}$$

$$\text{If } I_4 \neq \emptyset, T_3 \text{ is replaced by } T_3 + \sum_{i \in I_4} c_i X_{ij}, \text{ else } T_3 \text{ is unchanged,} \tag{3.31}$$

$$\text{If } I_4 \neq \emptyset, T_4 \text{ is replaced by } T_4 + \sum_{i \in I_4} X_{ij}^2, \text{ else } T_4 \text{ is unchanged,} \tag{3.32}$$

$$\text{If } I_2 \neq \emptyset, T_5 \text{ is replaced by } T_5 + \sum_{i \in I_2} c_i X_{ij}, \text{ else } T_5 \text{ is unchanged,} \tag{3.33}$$

$$\text{If } I_2 \neq \emptyset, T_6 \text{ is replaced by } T_6 + \sum_{i \in I_2} X_{ij}^2, \text{ else } T_6 \text{ is unchanged,} \tag{3.34}$$

$$\text{If } I_3 \neq \emptyset, T_7 \text{ is replaced by } T_7 - \sum_{i \in I_3} c_i X_{ij}, \text{ else } T_7 \text{ is unchanged,} \tag{3.35}$$

$$\text{If } I_3 \neq \emptyset, T_8 \text{ is replaced by } T_8 - \sum_{i \in I_3} X_{ij}^2, \text{ else } T_8 \text{ is unchanged.} \tag{3.36}$$

Next, to see how the sets are affected on the negative side, we define 4 sets in order to derive the following recursive relations:

1. SET 1: $I_1 = \{i \ : \ c_i X_{ij}^{-1} = -B_{-K_2}\}$.
2. SET 2: $I_2 = \{i \ : \ c_i X_{ij}^{-1} = -B_{-K_2-1}\}$.
3. SET 3: $I_3 = \{i \ : \ c_i X_{ij}^{-1} = B_{-K_2}\}$.
4. SET 4: $I_4 = \{i \ : \ c_i X_{ij}^{-1} = B_{-K_2-1}\}$.

Consider $F'(\cdot)$ on the negative side where $b_j < 0$. Let $K_2 = 1, ..., k_2$. For instance, when the knots change from B_{-K_2-1} to B_{-K_2}, $K_2 = 1, ..., k_2$, the index i is added to the set in (3.21) and T_1 increases by $c_i X_{ij}$ since the inequality, $c_i X_{ij}^{-1} > b_j$ is now loosened. Other sets are affected similarly leading to the following relations:

$$\text{If } I_4 \neq \emptyset, T_1 \text{ is replaced by } T_1 + \sum_{i \in I_4} c_i X_{ij}, \text{ else } T_1 \text{ is unchanged,} \tag{3.37}$$

$$\text{If } I_4 \neq \emptyset, T_2 \text{ is replaced by } T_2 + \sum_{i \in I_4} X_{ij}^2, \text{ else } T_2 \text{ is unchanged,} \tag{3.38}$$

$$\text{If } I_1 \neq \emptyset, T_3 \text{ is replaced by } T_3 - \sum_{i \in I_1} c_i X_{ij}, \text{ else } T_3 \text{ is unchanged,} \tag{3.39}$$

$$\text{If } I_1 \neq \emptyset, T_4 \text{ is replaced by } T_4 - \sum_{i \in I_1} X_{ij}^2, \text{ else } T_4 \text{ is unchanged,} \tag{3.40}$$

$$\text{If } I_3 \neq \emptyset, T_5 \text{ is replaced by } T_5 - \sum_{i \in I_3} c_i X_{ij}, \text{ else } T_5 \text{ is unchanged,} \tag{3.41}$$

$$\text{If } I_3 \neq \emptyset, T_6 \text{ is replaced by } T_6 - \sum_{i \in I_3} X_{ij}^2, \text{ else } T_6 \text{ is unchanged,} \tag{3.42}$$

$$\text{If } I_2 \neq \emptyset, T_7 \text{ is replaced by } T_7 + \sum_{i \in I_2} c_i X_{ij}, \text{ else } T_7 \text{ is unchanged,} \tag{3.43}$$

$$\text{If } I_2 \neq \emptyset, T_8 \text{ is replaced by } T_8 + \sum_{i \in I_2} X_{ij}^2, \text{ else } T_8 \text{ is unchanged.} \tag{3.44}$$

3.5 The FISCAL by the one-norm SVM

Instead of the squared SVM, we can consider the original SVM hinge loss function in the FISCAL. Then we should replace (3.3) with this followings:

$$\hat{\beta}_{j,m} \;=\; \arg\min_{b_{j,m}} \left\{ \sum_{i=1}^{n} [c_{i,m} - Y_i X_{ij} b_{j,m}]_+ + \lambda |b_{j,m}| \right\}, \tag{3.45}$$

where $\lambda > 0$ is a shrinkage parameter. Let

$$F(b_j) = \sum_{i=1}^{n} [c_i - Y_i X_{ij} b_j]_+ + \lambda |b_j|. \tag{3.46}$$

This objective function, $F(\cdot)$ is convex in b_j and is differentiable except at the knot points, $B_{-k_2} < ... < B_{-1} < 0 < B_1 < ... < B_{k_1}$. The derivative of the objective function is given by

$$F'(b_j) = -\sum_{i=1}^{n} Y_i X_{ij} I(c_i - Y_i X_{ij} b_j > 0) + \lambda \cdot \text{sign}(b_j), \text{ if } b_j \neq 0 \tag{3.47}$$

The first term in (3.47) can be simplified by separating into cases assuming that $Y_1, ..., Y_N = +1, Y_{N+1}, ..., Y_n = -1$ for simplification and computational savings:

1. For $(X_{ij} > 0)$

$$-\sum_{i=1}^{N} X_{ij} I(c_i X_{ij}^{-1} > b_j, X_{ij} > 0) \tag{3.48}$$

$$+ \sum_{i=N+1}^{n} X_{ij} I(c_i X_{ij}^{-1} > -b_j, X_{ij} > 0);$$

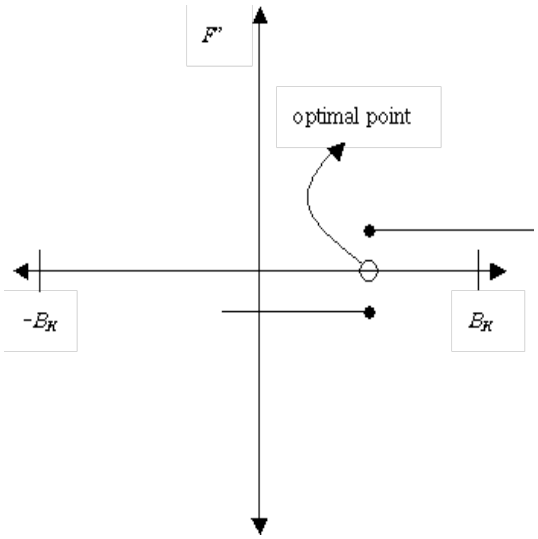

Figure 3.5: Jump derivative functions

2. For $(X_{ij} < 0)$

$$-\sum_{i=1}^{N} X_{ij} I(c_i X_{ij}^{-1} < b_j, X_{ij} < 0) \tag{3.49}$$

$$+\sum_{i=N+1}^{n} X_{ij} I(c_i X_{ij}^{-1} < -b_j, X_{ij} < 0);$$

The derivative of the objective function is an increasing flat function which is continuous except at the knot points. The derivative function jumps through zero from the negative side to the positive side only once. In addition, since the derivative function is dependent on the indicator functions, the above functional forms can change only at $b_j = 0$ or $b_j = \frac{c_i}{X_{ij}}$ where $i = 1, ..., N$, are observations for class $+1$ or $b_j = \frac{-c_i}{X_{ij}}$ where $i = N+1, ..., n$, are observations for class -1. This helps a lot to locate the minimum, since the derivative is flat in between these points. Let $B_{-k_2}, ..., B_{-1}, B_1, ..., B_{k_1}$ be distinct ordered values of $\frac{c_i}{X_{ij}}$ and $\frac{-c_i}{X_{ij}}$ where $B_{-k_2} < ... < B_{-1} < 0 < B_1 < ... < B_{k_1}$. Among the knots, we can locate the optimal point easily where the derivative function jumps through zero from the negative side to the positive side only once as shown in Figure 3.5.

A substantial savings in computing time of the FISCAL is possible by utilizing a recursive relation to relating the sums in (3.48)–(3.49).

Observe that

1. For $b_j > 0$,

$$F'(b_j) = -T_1 + T_2 - T_3 + T_4 + \lambda, \tag{3.50}$$

where

$$T_1 = \sum_{i=1}^{N} X_{ij} I(c_i X_{ij}^{-1} > b_j, b_j > 0, X_{ij} > 0), \tag{3.51}$$

$$T_2 = \sum_{i=N+1}^{n} X_{ij} I(c_i X_{ij}^{-1} > -b_j, b_j > 0, X_{ij} > 0), \tag{3.52}$$

$$T_3 = \sum_{i=1}^{N} X_{ij} I(c_i X_{ij}^{-1} < b_j, b_j > 0, X_{ij} < 0), \tag{3.53}$$

$$T_4 = \sum_{i=N+1}^{n} X_{ij} I(c_i X_{ij}^{-1} < -b_j, b_j > 0, X_{ij} < 0). \tag{3.54}$$

2. For $b_j < 0$,

$$F'(b_j) = -T_1 + T_2 - T_3 + T_4 - \lambda, \tag{3.55}$$

where

$$T_1 = \sum_{i=1}^{N} X_{ij} I(c_i X_{ij}^{-1} > b_j, b_j < 0, X_{ij} > 0), \tag{3.56}$$

$$T_2 = \sum_{i=N+1}^{n} X_{ij} I(c_i X_{ij}^{-1} > -b_j, b_j < 0, X_{ij} > 0), \tag{3.57}$$

$$T_3 = \sum_{i=1}^{N} X_{ij} I(c_i X_{ij}^{-1} < b_j, b_j < 0, X_{ij} < 0), \tag{3.58}$$

$$T_4 = \sum_{i=N+1}^{n} X_{ij} I(c_i X_{ij}^{-1} < -b_j, b_j < 0, X_{ij} < 0). \tag{3.59}$$

Basically, the recursive algorithm can be derived based on the natural inclusion-exclusion relations in the following 8 sets between the two consecutive knots where $B_{-k_2} < ... < B_{-1} < 0 < B_1 < ... < B_{k_1}$.

First, we define 4 sets in order to derive the following recursive relations:

1. SET 1: $I_1 = \{i \ : \ c_i X_{ij}^{-1} = B_{K_1}\}$.
2. SET 2: $I_2 = \{i \ : \ c_i X_{ij}^{-1} = B_{K_1 - 1}\}$.
3. SET 3: $I_3 = \{i \ : \ c_i X_{ij}^{-1} = -B_{K_1}\}$.
4. SET 4: $I_4 = \{i \ : \ c_i X_{ij}^{-1} = -B_{K_1 - 1}\}$.

Consider $F'(\cdot)$ on the positive side where $b_j > 0$. Let $K_1 = 1, ..., k_1$. For instance, when b_j changes from $B_{K_1 - 1}$ to B_{K_1}, $K_1 = 1, ..., k_1$, the index i drops from the set in (3.51) and T_1 decreases by $c_i X_{ij}$ since the inequality, $c_i X_{ij}^{-1} > b_j$ is now violated. Other sets are affected similarly leading to the following relations:

$$\text{If } I_1 \neq \emptyset, T_1 \text{ is replaced by } T_1 - \sum_{i \in I_1} X_{ij}, \text{ else } T_1 \text{ is unchanged,} \tag{3.60}$$

$$\text{If } I_4 \neq \emptyset, T_2 \text{ is replaced by } T_2 + \sum_{i \in I_4} X_{ij}, \text{ else } T_2 \text{ is unchanged,} \tag{3.61}$$

$$\text{If } I_2 \neq \emptyset, T_3 \text{ is replaced by } T_3 + \sum_{i \in I_2} X_{ij}, \text{ else } T_3 \text{ is unchanged,} \tag{3.62}$$

$$\text{If } I_3 \neq \emptyset, T_4 \text{ is replaced by } T_4 - \sum_{i \in I_3} X_{ij}, \text{ else } T_4 \text{ is unchanged.} \tag{3.63}$$

Second, we define 4 sets in order to derive the following recursive relations:

1. SET 1: $I_1 = \{i \ : \ X_{ij}^{-1} = -B_{-K_2}\}$.
2. SET 2: $I_2 = \{i \ : \ X_{ij}^{-1} = -B_{-K_2 - 1}\}$.

3. SET 3: $I_3 = \{i \ : \ X_{ij}^{-1} = B_{-K_2}\}$.

4. SET 4: $I_4 = \{i \ : \ X_{ij}^{-1} = B_{-K_2-1}\}$.

Consider $F'(\cdot)$ on the negative side where $b_j < 0$. Let $K_2 = 1, ..., k_2$. For instance, when b_j changes from B_{-K_2-1} to B_{-K_2}, $K_2 = 1, ..., k_2$, the index i is added to the set in (3.56) and T_1 increases by $c_i X_{ij}$ since the inequality, $c_i X_{ij}^{-1} > b_j$ is now loosened. Other sets are affected similarly leading to the following relations:

$$\text{If } I_4 \neq \emptyset, T_1 \text{ is replaced by } T_1 + \sum_{i \in I_4} X_{ij}, \text{ else } T_1 \text{ is unchanged,} \tag{3.64}$$

$$\text{If } I_1 \neq \emptyset, T_2 \text{ is replaced by } T_2 - \sum_{i \in I_1} X_{ij}, \text{ else } T_2 \text{ is unchanged,} \tag{3.65}$$

$$\text{If } I_3 \neq \emptyset, T_3 \text{ is replaced by } T_3 - \sum_{i \in I_3} X_{ij}, \text{ else } T_3 \text{ is unchanged,} \tag{3.66}$$

$$\text{If } I_2 \neq \emptyset, T_4 \text{ is replaced by } T_4 + \sum_{i \in I_2} X_{ij}, \text{ else } T_4 \text{ is unchanged.} \tag{3.67}$$

3.6 Simulation

In this section, we demonstrate the performance of the FISCAL in different settings. Since the FISCAL is designed for large p small n problems, we simulate the data from two high-dimensional sparse binary classification models, the supervised learning model and logistic regression model where $p > n$.

$$f : \mathbb{R}^p \to \{-1, 1\}, \boldsymbol{X}_i \in \mathbb{R}^p, i = 1, .., n,$$

where $(\boldsymbol{X_1}, Y_1), ..., (\boldsymbol{X_n}, Y_n) \in \mathbb{R}^p \times \{-1, 1\}$.

$$\log(\frac{p_i}{1 - p_i}) = \boldsymbol{X_i}^T \boldsymbol{\beta}, i = 1, .., n,$$

where

$$\mathrm{E}(Y_i | \boldsymbol{X}_i) = p_i = \frac{\exp(\boldsymbol{X_i}^T \boldsymbol{\beta})}{1 + \exp(\boldsymbol{X_i}^T \boldsymbol{\beta})},$$

Y_i is a 0/1 response, $\boldsymbol{X}_i^T$ is a $1 \times p$ vector for predictors, and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ is a vector for parameters.

The simulated data consist of a training set, a validation set, and an independent test set of size 100. We assume that the training and tuning sets have the same sample size n. For each method, we fit the model using the training data and use the validation set to select the tuning parameter. The misclassification error on the test set is used to evaluate prediction performance of each method. We run 50 simulations for each experiment and report the average results. We implement the FISCAL in R. For comparison, we also consider the one-norm SVM by the NLPSVM, the SCAD SVM by the successive quadratic algorithm and the penalized logistic regression by the coordinate descent algorithm. The NLPSVM and successive quadratic algorithm are available in MATLAB and the coordinate-wise descent algorithm available in R. The optimal tuning parameter for each method is chosen by a grid search using the validation set.

3.6.1 Simulation settings

We present ten examples, which consider different sample sizes, different predictor sizes and correlation scenarios among $\boldsymbol{X}$. Here are the details of ten examples.

(a) In Example 1, we have $p = 200$ and $n = 20$. Based on the Weston et al.'s supervised learning model, the first three covariates, X_1, X_2 and X_3, are important and the rest 197 predictors are redundant [31]. Concretely speaking, we have $X_1 = YN(0.5, 1)$, $X_2 = YN(0.5, 1)$, $X_3 = YN(0.5, 1)$ and X_4 to X_{200} are i.i.d. $N(0, 20)$. The values $\epsilon = 0.1$ and $M = 3$ are used.

(b) Example 2 is the same as Example 1, except that $n = 50$. The values $\epsilon = 0.1$ and $M = 3$ are used.

(c) In Example 3, we have $p = 200$ and $n = 20$. Based on the logistic regression model, $X_1, \ldots, X_p$ are i.i.d. from $N(0, 1)$. $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ can consists of the first 3 non-zero coefficients, (3, 6, 9) and the rest 197 zero coefficients. In other words, β_1 is 3, β_2 is 6, β_3 is 9 and $\beta_4 - \beta_{200}$ are 0. The values $\epsilon = 0.1$ and $M = 3$ are used.

(d) Example 4 is the same as Example 3, except that there is moderate correlation among all covariates. In particular, the pairwise correlation between any two covariates X_i and X_j is $\text{corr}(X_i, X_j) = \rho^{|i-j|}$ where $\rho = 0.5$. The values $\epsilon = 0.1$ and $M = 3$ are used.

(e) Example 5 is the same as Example 3, except that there is high correlation among all covariates. In particular, the pairwise correlation between any two covariates X_i and X_j is $\text{corr}(X_i, X_j) = \rho^{|i-j|}$ where $\rho = 0.9$. The values $\epsilon = 0.1$ and $M = 3$ are used.

(f) Example 6 is the same as Example 3, except that $\text{E}(Y_i|\boldsymbol{X}_i) = p_i = \Phi(\exp(\boldsymbol{X_i}^T\boldsymbol{\beta}))$. The values $\epsilon = 0.1$ and $M = 3$ are used.

(g) Example 7 is the same as Example 6, except that there is moderate correlation among all covariates. In particular, the pairwise correlation between any two covariates X_i and X_j is $\text{corr}(X_i, X_j) = \rho^{|i-j|}$ where $\rho = 0.5$. The values $\epsilon = 0.1$ and $M = 3$ are used.

(h) Example 8 is the same as Example 6, except that there is high correlation among all covariates. In particular, the pairwise correlation between any two covariates X_i and X_j is $\text{corr}(X_i, X_j) = \rho^{|i-j|}$ where $\rho = 0.9$. The values $\epsilon = 0.1$ and $M = 3$ are used.

(i) Example 9 is the same as Example 4, except that $p = 1000$. The values $\epsilon = 0.1$ and $M = 2$ are used.

(j) Example 10 is the same as Example 9, except that $n = 50$. The values $\epsilon = 0.1$ and $M = 2$ are used.

3.6.2 Experiments and results

We compare 5 different methods with regard to their misclassification error and variable selection performance. In particular, the misclassification error on the test set,

$$\frac{\sum_{i=1}^{100} I(Y_i \neq \hat{f}(\boldsymbol{X}_i))}{100},$$

where

$$\hat{f}(\boldsymbol{X}_i) = \text{sign}(\boldsymbol{X}_i^T\hat{\boldsymbol{\beta}}),$$

Table 3.1: Simulation results for Example 1

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.41 (0.09)	2.12	2.12
FISCAL (One-norm SVM)	0.43 (0.10)	2.40	2.38
One-norm SVM	0.42 (0.09)	1.50	29.76
SCAD SVM	0.40 (0.08)	0.96	60.02
Penalized LR	0.38 (0.08)	0.70	43.48

Table 3.2: Simulation results for Example 2

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.31 (0.08)	1.30	1.30
FISCAL (One-norm SVM)	0.33 (0.10)	1.60	1.60
One-norm SVM	0.37 (0.09)	0.62	40.50
SCAD SVM	0.30 (0.08)	0.68	14.72
Penalized LR	0.29 (0.06)	0.58	15.98

is used to evaluate their prediction performance. For variable selection, we report two types of selection errors. Selection error I is defined as the number of non-zero coefficients which are estimated as zero, and selection error II is the number of zero coefficients which are not estimated as zero. Generally speaking, it is observed that the FISCAL by both the squared SVM and the one-norm SVM gives quite competitive performance compared with the other algorithms, when p is larger than n. In particular, the FISCAL produced the minimum misclassification error in Examples 3–10 leading to more parsimonious models where the logistic regression model and correlated predictors are considered. We guess that throughout Examples 1–10, the FISCAL is the best in terms of selection error II because M used in the FISCAL is at most 3, while the penalized logistic regression is the best in terms of selection error I because the log-likelihood is used for the penalized logistic regression. We mainly focus on test error.

Tables 3.1 and 3.2 summarize the results for Examples 1 and 2. In Examples 1 and 2, the penalized logistic regression works best in terms of the test error. Tables 3.3–3.5 represent the results for Examples 3–5. The FISCAL by the squared SVM is the best

Table 3.3: Simulation results for Example 3

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.286 (0.13)	1.70	1.70
FISCAL (One-norm SVM)	0.293 (0.14)	1.82	1.82
One-norm SVM	0.36 (0.10)	1.38	19.92
SCAD SVM	0.33 (0.11)	1.14	17.52
Penalized LR	0.29 (0.09)	0.92	18.30

Table 3.4: Simulation results for Example 4

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.20 (0.09)	1.54	1.54
FISCAL (One-norm SVM)	0.19 (0.09)	1.58	1.50
One-norm SVM	0.29 (0.10)	1.10	26.92
SCAD SVM	0.25 (0.10)	1.26	14.74
Penalized LR	0.22 (0.08)	0.62	14.60

Table 3.5: Simulation results for Example 5

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.14 (0.07)	1.72	1.62
FISCAL (One-norm SVM)	0.13 (0.06)	1.62	1.40
One-norm SVM	0.21 (0.08)	1.16	10.40
SCAD SVM	0.16 (0.07)	1.68	3.26
Penalized LR	0.17 (0.08)	0.42	20.02

Table 3.6: Simulation results for Example 6

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.286 (0.13)	1.70	1.70
FISCAL (One-norm SVM)	0.293 (0.14)	1.82	1.82
One-norm SVM	0.36 (0.10)	1.38	19.92
SCAD SVM	0.33 (0.11)	1.14	17.52
Penalized LR	0.34 (0.08)	0.94	36.28

Table 3.7: Simulation results for Example 7

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.20 (0.09)	1.54	1.54
FISCAL (One-norm SVM)	0.19 (0.09)	1.58	1.50
One-norm SVM	0.29 (0.10)	1.10	26.92
SCAD SVM	0.25 (0.10)	1.26	14.74
Penalized LR	0.24 (0.08)	0.74	13.06

Table 3.8: Simulation results for Example 8

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.14 (0.07)	1.72	1.62
FISCAL (One-norm SVM)	0.13 (0.06)	1.62	1.40
One-norm SVM	0.21 (0.08)	1.16	10.40
SCAD SVM	0.16 (0.07)	1.68	3.26
Penalized LR	0.19 (0.08)	0.46	21.12

Table 3.9: Simulation results for Example 9

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.263 (0.15)	2.00	1.00
FISCAL (One-norm SVM)	0.258 (0.16)	1.96	0.96
One-norm SVM	0.36 (0.10)	1.50	47.10
SCAD SVM	0.30 (0.11)	1.22	76.94
Penalized LR	0.29 (0.11)	0.72	42.36

Table 3.10: Simulation results for Example 10

Method	Test Error	Selection Error I	Selection Error II
FISCAL (Squared SVM)	0.08 (0.06)	1.08	0.08
FISCAL (One-norm SVM)	0.10 (0.06)	1.20	0.20
One-norm SVM	0.17 (0.07)	0.48	22.20
SCAD SVM	0.12 (0.06)	0.74	4.84
Penalized LR	0.11 (0.05)	0.30	13.74

with regard to the test error in Example 3 whereas the FISCAL by the one-norm SVM is in Examples 4–5. In fact, it should be noted that the penalized logistic regression has the gifted advantage of assuming the correct data generating procedure in Examples, 3–5. That is why we consider the probit link function,

$$\mathrm{E}(Y_i|\boldsymbol{X}_i) = p_i = \Phi(\exp(\boldsymbol{X_i}^T\boldsymbol{\beta}))$$

instead of the logit link function,

$$\mathrm{E}(Y_i|\boldsymbol{X}_i) = p_i = \frac{\exp(\boldsymbol{X_i}^T\boldsymbol{\beta})}{1+\exp(\boldsymbol{X_i}^T\boldsymbol{\beta})}$$

in Examples 6–8. Similarly, the FISCAL by the squared SVM is the best with regard to the test error in Example 6, whereas the FISCAL by the one-norm SVM is in Examples 7–8. Noticeably, the prediction performance of the penalized logistic regression get a little worse by the change of the link function. The results are shown in Tables 3.6–3.8. We guess that throughout Tables 3.3–3.8, the squared hinge loss function may be good for the independent covariates, whereas the original hinge loss function may be good for the correlated covariates. When correlation increases, even unimportant variables may be also helpful in explaining the binary response. That's why the test error may improve as correlation increases in Examples 3–8. The FISCAL by the one-norm SVM is the winner in terms of the test error in the case of $p = 1000$ and $n = 20$, while the FISCAL by the squared SVM shows the best performance in the case of $p = 1000$ and $n = 50$. The results can be observed in Tables 3.9–3.10. Expectedly, the performance of all the methods get better when the sample size increases and get worse when the number of redundant predictors increases.

3.7 Real data example

We consider the gene expression data used in Bühlmann (2006) [9]. There are 7129 gene expressions and a binary response describing the status of lymph node involvement in breast cancer in 49 breast tumor samples. For a prescreening, we select only 265 expressions with the largest variances from 7129 expressions. That is, we consider the 265 predictors and the binary response which represents a high dimensional sparse binary classification model. Most significantly, the 265 predictors are significantly correlated. We apply the FISCAL with $\epsilon = 0.1$ and $M = 2$ using the recursive algorithm, the one-norm SVM by the

Table 3.11: Real example results

Method	Test Error	Number of selected variables
FISCAL (Squared SVM)	0.22	2
FISCAL (One-norm SVM)	0.22	2
One-norm SVM	0.33	19
SCAD SVM	0.33	19
Penalized LR	0.56	260

NLPSVM, the SCAD SVM by the successive quadratic algorithm and the penalized logistic regression by the coordinate-wise descent algorithm to the high dimensional sparse binary classification model with the correlated predictors. We randomly select 40 training data and 9 test data. For tuning parameters, we used a grid search. Then 5 fold cross-validation is conducted with 40 training data in R . Test error and the number of non-zero estimates are shown in Table 3.11. Similarly, the FISCAL generally gives more parsimonious models compared to the one-norm SVM, SCAD SVM and penalized logistic regression providing better prediction power as well.

3.8 Discussion

We suggest a new variable selection algorithm for high dimensional sparse binary classification models and the recursive algorithm for computational reduction. First, we suggest the squared SVM where both one-norm penalty and two-norm penalty are considered at the same time. The FISCAL by the squared SVM is a combination of one dimensional squared SVM and forward selection. More precisely, the FISCAL by the squared SVM takes an advantage of nearly closed form solution obtained by calculating derivatives in the one-dimensional squared SVM and then use forward selection to fit models on the errors repeatedly. Then the algorithm uses piecewise linearity and continuity to search for zeros, by evaluating the derivative function at the knot points. The recursive algorithm relates the derivative function at knots, which leads to substantial savings in computation time. Second, we apply the processes to the FISCAL using the original one-norm SVM instead of the squared SVM. Throughout our simulation study and real data example, we find that the FISCAL by both the squared SVM and the one-norm SVM generally shows good prediction

performance and selection accuracy in comparison with the one-norm SVM, SCAD SVM and penalized logistic regression. As a future work, we may have to make an effort for more computational savings.

Bibliography

[1] Akaike H. (1973) *Maximum likelihood identification of gaussian autoregressive moving average models.* Biometrika, No. 60, pp. 255–265.

[2] Bazaraa M. and Shetty C. (1979) *Nonlinear programming.* John Wiley.

[3] Bennett K. and Blue J. (1997) *A support vector machine approach to decision trees.* Department of Mathematical Sciences Math Report No. 97-100, Rensselaer Polytechnic Institute, Troy, NY, 12180, 1997. http://www.math.rpi.edu/bennek/.

[4] Bennett K. and Bredensteiner E. (2000) *Geometry in learning.* Mathematical Association of America, pp. 132–145.

[5] Bennett K. and Bredensteiner E. (2000) *Support Vector Machines: Hype or Hallelujah?.* SIGKDD Explorations, Vol. 2, Issue 2, pp. 1–13.

[6] Bondell H. and Reich B. (2008) *Simultaneous regression shrinkage, variable selection and clustering of predictors with OSCAR.* Biometrics 64, pp. 115–123.

[7] Bradley P. S. and Mangasarian O. L. (1998) *Feature selection via concave minization and support vector machines.* Proceeding 15th International Conference on Machine Learning, pp. 82–90.

[8] Breiman L. (1995) *Better subset selection using the non-negative garotte.* Technometrics, Vol. 37, No. 4, pp. 373–384.

[9] Bühlmann P. (2006) *Boosting for high-dimensional linear models.* Annals of Statistics, No. 34, pp. 559–583.

[10] Bühlmann P. and Hothorn T. (2007) *Boosting Algorithms: Regularization, Prediction, and Model Fitting.* Statistical Science, Vol. 22, No. 4, pp. 447–505.

[11] Bühlmann P. and Yu B. (2005) *Boosting, model selection, lasso and nonnegative garrote.* Technical report, ETH Zrich.

[12] Bühlmann P. and Yu B. (2006) *Sparse Boosting.* Journal of Machine Learning Research, No. 7, pp. 1001–1024.

[13] Cortes C. and Vapnik V. (1995) *Support vector networks.* Machine Learning, 20, pp. 273–279.

[14] Efron B., Hastie T., Johnstone I. and Tibshirani R. (2004) *Least Angle Regression.* Annals of Statistics, No. 24, pp. 407–499.

[15] Fan J. and Li R. (2001) *Variable selection via penalized likelihood.* Journal of the American Statistical Association, No. 96, pp. 1348–1360.

[16] Friedman J. (1999) *Stochastic gradient boosting.* Technical report, Stanford University.

[17] Friedman J., Hastie T., Hofling H. and Tibshirani R. (2007) *Pathwise Coordinate Optimization.* Annals of Applied Statistics, Vol.1, No. 2, pp. 302–332.

[18] Friedman J., Hastie T. and Tibshirani R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent.* Department of Statistics, Stanford University.

[19] Fung G. and Mangasarian O. L. (2004) *A feature selection newton method for support vector machine classification.* Computational Optimization and Applications Journal, 28(2), pp. 185–202.

[20] Gill E., Murray W., and Saunders A. (2005) *a Fortran package for large-scale linear and quadratic programming.* Technical Report NA 05-1, Department of Mathematics, University of California, San Diego.

[21] Hastie T., Tibshirani R., and Friedman J. (2001) *The Elements of Statistical Learning.* Springer Series in Statistics.

[22] Hoerl E. and Kennard W. (1970) *Ridge Regression: biased estimation for nonorthogonal problems.* Technomerics, No. 12, pp. 55–67.

[23] Huang J., Ma S. and Zhang C.-H. (2008) *Adaptive Lasso for sparse high-dimensional regression models.* Statisica Sinica, No. 18, pp. 1603–1618.

[24] Lawson C. and Hanson R. (1974) *Solving Least Squares Problems.* Englewood Cliffs, NJ:Prentice-Hall.

[25] Rawlings J., Pantula S. and Dickey D. (2001) *Applied Regression Analysis.* Springer.

[26] Scholkorf B., Burges C. and Smola J. (1999) *Advances in Kernel methods: Support Vector Learning.* Massachusetts Institute of Technology.

[27] Schwarz G. (1978) *Estimation the dimension of a model.* Annals of Statistics, No. 6, pp. 461–464.

[28] Tibshirani R. (1996) *Regression Shrinkage and Selection via the Lasso.* Journal of Royal Statistical Society, No. 1, pp. 147–169.

[29] Wang L. Zhu J. and Zou H. (2006) *The Doubly Regularized Support Vector Machine.* Statistica Sinica, 16(2), pp. 589–616.

[30] Wang L. Zhu J. and Zou H. (2008) *Hybrid huberized support vector machines for microarray classification and gene selection.* Bioinformatics, 24(3), pp. 412–419.

[31] Weston J., Mukherjee S., Chapelle O., Pontil M., Poggio T. and Vapnik V. (2000) *Feature selection for SVMs.* Advances in Neural Information Processing Systems, 13, pp. 668–674.

[32] Zhang H., Ahn J., Lin X. and Park C. (2006) *Gene selection using support vector machines with non-convex penalty.* Bioinformatics, 22(1), pp. 88–95.

[33] Zhao P. and Yu B. (2006) *On Model Selection Consistency of Lasso.* Journal of Machine Learning Research, No. 7, pp. 2541–2563.

[34] Zhao P. and Yu B. (2007) *Stagewise LASSO.* Journal of Machine Learning Research, No. 8, pp. 2701-2726.

[35] Zou H. (2006) *The Adaptive Lasso and Its Oracle Properties.* Journal of the American Statistical Association, No. 476, pp. 1418–1429.

[36] Zou H. and Hastie T. (2005) *Regularization and variable selection via the elastic net.* Journal of Royal Statistical Society, No. 67, Part 2, pp. 301–320.

[37] Zou H. and Zhang H. (2009) *On The Adaptive Elastic-Net With a Diverging Number of Parameters.* Annals of Statistics, 37, pp. 1733–1751.