PREDICTION AND VARIABLE SELECTION IN SPARSE ULTRAHIGH DIMENSIONAL ADDITIVE MODELS

by

GIRLY MANGUBA RAMIREZ

B.S., University of the Philippines Diliman, 2001

M.S., University of the Philippines Los Banos, 2008

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the

requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Statistics

College of Arts and Sciences

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2013

Abstract

The advance in technologies has enabled many fields to collect datasets where the number of covariates (p) tends to be much bigger than the number of observations (n), the so-called ultrahigh dimensionality. In this setting, classical regression methodologies are invalid. There is a great need to develop methods that can explain the variations of the response variable using only a parsimonious set of covariates. In the recent years, there have been significant developments of variable selection procedures. However, these available procedures usually result in the selection of too many false variables. In addition, most of the available procedures are appropriate only when the response variable is linearly associated with the covariates. Motivated by these concerns, we propose another procedure for variable selection in ultrahigh dimensional setting which has the ability to reduce the number of false positive variables. Moreover, this procedure can be applied when the response variable is continuous or binary, and when the response variable is linearly or non-linearly related to the covariates. Inspired by the Least Angle Regression approach, we develope two multi-step algorithms to select variables in sparse ultrahigh dimensional additive models. The variables go through a series of nonlinear dependence evaluation following a Most Significant Regression (MSR) algorithm. In addition, the MSR algorithm is also designed to implement prediction of the response variable. The first algorithm called MSR-continuous (MSRc) is appropriate for a dataset with a response variable that is continuous. Simulation results demonstrate that this algorithm works well. Comparisons with other methods such as greedy-INIS by Fan et al. (2011) and generalized correlation procedure by Hall and Miller (2009) showed that MSRc not only has false positive rate that is significantly less than both methods, but also has accuracy and true positive rate comparable with greedy-INIS. The second algorithm called MSR-binary (MSRb) is appropriate when the response variable is binary. Simulations demonstrate that MSRb is competitive in terms of prediction accuracy and true positive rate, and better than GLMNET in terms of false positive rate. Application of MSRb to real datasets is also presented. In general, MSR algorithm usually selects fewer variables while preserving the accuracy of predictions.

KEY WORDS: Additive model; Smoothing; Sparsity; Ultrahigh dimensional; Variable selection.

PREDICTION AND VARIABLE SELECTION IN SPARSE ULTRAHIGH DIMENSIONAL ADDITIVE MODELS

by

GIRLY MANGUBA RAMIREZ

B.S., University of the Philippines Diliman, 2001

M.S., University of the Philippines Los Banos, 2008

A DISSERTATION

submitted in partial fulfillment of the

requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Statistics

College of Arts and Sciences

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2013

Approved by:

Major Professor

Haiyan Wang

Abstract

The advance in technologies has enabled many fields to collect datasets where the number of covariates (p) tends to be much bigger than the number of observations (n), the so-called ultrahigh dimensionality. In this setting, classical regression methodologies are invalid. There is a great need to develop methods that can explain the variations of the response variable using only a parsimonious set of covariates. In the recent years, there have been significant developments of variable selection procedures. However, these available procedures usually result in the selection of too many false variables. In addition, most of the available procedures are appropriate only when the response variable is linearly associated with the covariates. Motivated by these concerns, we propose another procedure for variable selection in ultrahigh dimensional setting which has the ability to reduce the number of false positive variables. Moreover, this procedure can be applied when the response variable is continuous or binary, and when the response variable is linearly or non-linearly related to the covariates. Inspired by the Least Angle Regression approach, we develope two multi-step algorithms to select variables in sparse ultrahigh dimensional additive models. The variables go through a series of nonlinear dependence evaluation following a Most Significant Regression (MSR) algorithm. In addition, the MSR algorithm is also designed to implement prediction of the response variable. The first algorithm called MSR-continuous (MSRc) is appropriate for a dataset with a response variable that is continuous. Simulation results demonstrate that this algorithm works well. Comparisons with other methods such as greedy-INIS by Fan et al. (2011) and generalized correlation procedure by Hall and Miller (2009) showed that MSRc not only has false positive rate that is significantly less than both methods, but also has accuracy and true positive rate comparable with greedy-INIS. The second algorithm called MSR-binary (MSRb) is appropriate when the response variable is binary. Simulations demonstrate that MSRb is competitive in terms of prediction accuracy and true positive rate, and better than GLMNET in terms of false positive rate. Application of MSRb to real datasets is also presented. In general, MSR algorithm usually selects fewer variables while preserving the accuracy of predictions.

KEY WORDS: Additive model; Smoothing; Sparsity; Ultrahigh dimensional; Variable selection.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I give thanks and praises to God for His marvelous work in my life during my five years of studies at Kansas State University. With that, I thank my major professor Dr. Haiyan Wang for her persistent efforts in guiding and helping me with this dissertation. My appreciation also to the committee members: Dr. Weixin Yao, Dr. Christopher Vahl, Dr. Zhijian Pei, Dr. Paul Nelson and the chair of the committee from the Math department, Dr. David Yetter. I also thank the faculty and staff of the Department of Statistics who have been very helpful throughout my studies at K-State. Thank you also to all graduate students for their friendship. Last but not the least, my sincere appreciation goes to my family members: Noel, Jaron, Mama, Papa, Manang Grace, Manung Jon, Ading Cj, Auntie Mercy and Lola for their love, encouragement and prayer support.

Chapter 1

Introduction

Given the remarkable advance in technologies and computing power, researchers are now capable of collecting very large and complex datasets. Some examples of these data are found in atmospheric science, microarrays, genomics, information technology, biogeochemical and large-scale e-commerce. In these examples, the number of covariates (p) tends to grow much faster than the number of observations (n), the so-called nonpolynomial (NP) dimensionality or ultrahigh dimensionality. The need to analyze these kinds of dataset poses a great challenge for those in the fields of statistics and machine learning. In recent years, there have been significant developments in the analysis of these datasets. But, there remains a great deal to do.

A typical problem in statistical inference is to select a parsimonious set from a large collection of covariates for the efficient prediction of the response. That is, suppose that we have a random sample $(\mathbf{X}_i, Y_i)$, $i = 1, \ldots n$ from the population

$$Y = m(\mathbf{X}) + \varepsilon \tag{1.0.1}$$

in which $\mathbf{X}_i = (X_{i1}, \ldots, X_{ip})^T$ are random variables, ε is a random error with conditional mean zero and $cov(X, \varepsilon) = \mathbf{0}$. The main objective is to estimate $m(\mathbf{X})$ by minimizing the following function $\Sigma_{i=1}^{n}(Y_i - \widehat{m}(X_i))^2$, where $\widehat{m}(X_i)$ is an additive function of the components of the argument. Or suppose that Y_i is binary and the relationship between Y_i and $\mathbf{X}_i$ is described by the following nonlinear equation:

$$E(Y) = \frac{e^{m(\mathbf{X})}}{1 + e^{m(\mathbf{X})}} \qquad (1.0.2)$$

In this case, the main objective is to estimate $m(\mathbf{X})$ such that the classification error is minimized, where $\widehat{m}(\mathbf{X})$ is an additive function of the components of the argument.

Dimension reduction plays a vital role in NP or ultrahigh dimensional problems with the consideration of the sparsity assumption, which assumes that among the large number of independent variables, only a small set is related to the response variable. There exist numerous statistical procedures for data reduction and these include LARS by Efron et al. (2004), SCAD by Fan and Li (2001), and Dantzig selector by Candes and Tao (2007). However, these penalized methods are difficult to apply to ultrahigh dimensional datasets due to challenges in computational expediency, statistical accuracy and algorithmic stability. Fan and Lv (2008) proposed a two-stage screening method in which they first perform dimension reduction of the model and then apply penalized methods. This screening method called Sure Independence Screening (SIS) is successful in overcoming the aforementioned challenges. A great feature of SIS is that the dimensionality of the model is allowed to grow exponentially in the sample size. In addition, SIS requires normality of the response variable and is designed specifically for linear models. Fan and Lv (2008) also extended SIS to cover cases when the regularity conditions fail, and this methodological extension is called Iterated Sure Independence Screening (ISIS). Fan et al. (2009) developed two possible variants of SIS and ISIS that have attractive theoretical properties in terms of reducing the false selection rates. However, SIS-based screening methods are based on correlation which assumes that the response variable and covariates are normally distributed, and that the relationship of the covariates with the response is linear. Thus, SIS-based procedures are said to be methodologically challenged when the covariates are not jointly normal and when the marginal or joint regression of the covariates with the response is nonlinear. Thus, procedures for variable selection in nonparametric modeling is essential.

Procedures on variable selection in nonparametric modeling is limited. In practice, there is usually not enough prior information that the effects of the covariates take a linear form or belong to any parametric family. Sometimes, substantial improvements are possible by using a more flexible class of nonparametric models, such as the additive model $Y = \sum_{j=1}^{p} m_j(X_j) + \varepsilon$, introduced by Stone (1985). Use of an additive model substantially improves the flexibility of the ordinary linear model and allows transformed covariates to enter into the linear model. At present, the literature on variable selection utilizing nonparametric additive models is limited. Many of the available procedures are extensions of LASSO such as the sparse additive models (SpAM) by Ravikumar et al. (2009) and COSSO which is developed by Lin and Zhang (2006). Another procedure is the work of Huang et al. (2010) which is an extension of adaptive LASSO to additive models. A penalty that combines sparsity and smoothness with a fixed design was proposed by Meier et al. (2009). Unfortunately, all of these procedures are extensions of penalized pseudolikelihood approaches to additive modeling, and hence, still suffer from the aforementioned three challenges in NP dimensional settings. A most recent screening method, Nonparametric Independence Screening (NIS), was developed by Fan et al. (2011) which is a variation of SIS based on nonparametric marginal regression. Fan et al. (2011) further improved the NIS procedure by developing the iterative NIS (INIS) and greedy-INIS. NIS-based methods consider correlation learning by ranking the magnitude of marginal estimators, nonparametric marginal correlations, and the marginal residual sum of squares. That is, one marginal nonparametric regression of the response Y are fitted against each covariate X_i separately and the importance of each covariate to the joint model is based on a measure of the goodness of fit of their marginal models. The magnitude of these marginal utilities can preserve the non-sparsity of the joint additive models under some reasonable conditions, even with converging minimum strength of signals. NIS methods are two-stage procedures and can deal with the aforementioned three challenges better than the other methods. Nevertheless, NIS-based procedures have high false selection rate when the covariates are correlated with each other. This is because NIS-based procedures assume that the active covariates are independent of the nonactive covariates.

The procedures that have been discussed are appropriate when the response variable is continuous. Variable selection and classification in datasets with a binary response variable is also very relevant to many fields such as in bioinformatics and image recognition. In bioinformatics, ultrahigh dimensional gene expression datasets are very common. For example, scientists want to know which of these genes have strong contribution to the binary response variable which assumes two values, namely, occurrence and non-occurrence of an event. Three of the current popular methods are Generalized Linear Models with Elastic Net (GLMNET), Binary Matrix Shuffling Filter (BMSF) and Gene Selection in Random Forest (GeneSrF). The procedure GLMNET is able to do variable selection and classification simultaneously. On the other hand, BMSF and GeneSrF conducts variable selection. The latter two procedures are combined with classification techniques such as Support Vector machine (SVM), Naive Bayes (NB), linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) to perform classification of new observations. According to literature, the aforementioned procedures have very good performance in terms of accuracy in classifying new observations. However, GLMNET, BMSF and GeneSrF have the tendency to select many variables, which indicates that they also have the tendency to select too many false positive variables.

In general, existing procedures for both continuous and binary response variables may have the problem of selecting too many false positive variables. Motivated by this concern, this paper proposes the Most Significant Regression (MSR) algorithm which can be used for variable selection and prediction. With the high false selection rates in existing procedures, MSR algorithm aims to reduce the false selection rates.

MSR algorithm relates to the Least Angle Regression (LARS) by Efron et al. (2004) but taking into consideration the possibility of nonlinear relationships between the response and the covariates. In addition, MSR, unlike the NIS-based procedures, BMSF and GeneSrF, simultaneously conducts variable selection and response estimation. As demonstrated in

Monte Carlo simulations, results of MSR when the response variable is continuous resulted in smaller false selection rates and better predictive ability than NIS-based procedures. In addition, results of MSR when the response variable is binary showed that MSR always selects fewer variables than existing procedures while maintaining comparable classification accuracy.

Chapter 2

Literature Review

In the last decade, large data sets with large numbers of variables are more and more common. This has stimulated the development of procedures that can perform variable selection and data reduction on large data sets. Specifically, reviews of variable selection procedures and prediction techniques are discussed in this chapter.

2.1 Problems of ultrahigh dimensional setting

Richard Bellman (1961) coined the term "curse of dimensionality" which in statistics, refers to the issues caused by the rapid increase in the number of covariates p given a fixed sample size n. Due to the curse of dimensionality, the data are very scattered and thus, it is quite difficult to achieve accurate predictions of the response. With a very large number of covariates, unnecessary predictors may be present and will add noise to the estimation of the response. Collinearity among the predictors is also likely to exist. In addition, with very large p, the computational cost in model building is very expensive.

To avoid the curse of dimensionality, several techniques have been proposed and two of these are variable selection and additive modeling. The main goal in variable selection is to select the best subset of covariates. "Best" refers to a parsimonious model that has small sum of square error, or large adjusted R^2, or low prediction error and other criteria available in literature. Selection of the best subset of covariates can improve significantly the computational cost in estimation. On the other hand, additive modeling which was introduced by Stone (1985) can significantly improve the flexibility of the variable selection procedure.

2.2 Continuous Case

2.2.1 Variable Selection for Parametric Models

In this section, procedures for variable selection in parametric models, those that assume a linear relationship between response and predictors are discussed. Initiated by Donoho and Johnstone (1994), and following the Least Absolute Shrinkage and Selection Operator (LASSO) by Tibshirani (1996), many penalized pseudo-likelihood procedures and related methods have been studied in the literature in the setting of parametric models assuming a linear or generalized linear relationship between the response and predictors. A recent advance in ultrahigh dimensional variable selection is the development of screening methods which are deemed better than the penalized procedures in terms of statistical accuracy, computational expediency and algorithmic stability. Procedures for variable selection in the parametric setting are discussed in two separate sections, namely penalized methods and screening methods.

Penalized Methods

In high dimensional statistical endeavors, the purpose of applying the penalized methods is to simultaneously select variables and estimate the regression coefficients by maximizing the following penalized likelihood function:

$$n^{-1}l_n(\beta) - \sum_{j=1}^{p} p_\lambda(|\beta_j|) \tag{2.2.1}$$

where $l_n(\beta)$ is the assumed log-likelihood and $p_\lambda(\cdot)$ is a penalty function indexed by $\lambda \geq 0$, a regularization parameter. Variables with associated estimated regression coefficients equal to zero are deleted.

Fan and Li (2001) support a penalty function that produce estimators that have the following properties: sparsity, unbiasedness and continuity. Sparsity implies that the estimator sets small coefficients to zero, therefore reducing the complexity of the model. For the unbiasedness property, the estimator derived from the penalty function is said to be nearly unbiased when the true parameter $|\beta_j|$ is large. For continuity, the resulting estimator is continuous in the data to increase stability in model prediction. The following are the penalized methods developed for variable selection in ultrahigh dimensional and parametric setting.

Ridge Regression, HoerlA.E. and R.W. (1970). The main task of ridge regression is to find a linear function that models the relationships between a continuous response variable and continuous covariates. In ridge regression, the goal is to minimize the residual sum of squares (RSS) subject to a constraint of the form $\Sigma|\beta_j|^2 \leq t$. It yields an estimator for the regression coefficients equal to $\widehat{\beta} = (X^T X + \lambda I)^{-1} X^T y$. When the covariates are highly correlated, ridge regression is able to restrain the size of the estimated regression coefficients by including a penalty which reduces the undesirable symptoms of correlated covariates. Using this formulation, ridge regression may be seen as a penalized L_2 -regression in which $p_\lambda(|\theta|) = \lambda|\theta|^2$.

Bridge Regression, Frank and Friedman (1993). Bridge regression minimizes the RSS subject to a constraint $\Sigma|\beta_j|^q \leq t$. This procedure is a penalized L_q-regression, a natural generalization of penalized L_0-regression in which $p_\lambda(|\theta|) = \lambda|\theta|^q$ for $0 < q \leq 2$. This bridges the best subset selection (penalized L_0) and ridge regression (penalized L_2), including the L_1-penalty as a specific case.

Least absolute shrinkage and selection operator (LASSO), Tibshirani (1996) . LASSO minimizes RSS subject to a constraint $\Sigma|\beta_j| \leq t$. The LASSO is also known as the penalized L_1-regression in the ordinary regression setting, in which $p_\lambda(|\theta|) = \lambda|\theta|$. The selected model in LASSO fits the mean $X\beta$ well if its bias

$$Bias = \|(I - \widehat{P})X\beta\|$$

is small. $\widehat{P}$ is the projection to the linear span of the set of selected variables and I is the $n \times n$ identity matrix. When θ is large, the LASSO estimator has a bias approximately of size $\lambda \geq 0$, where λ is the regularization parameter index of the penalty function. As a result, the LASSO estimator has to choose a smaller λ in order to compensate the bias problem and obtain a desired mean squared error. However, a smaller value of λ results in a complex model. This explains why the LASSO estimator tends to have many false positive variables in the selected model.

Smoothly clipped absolute deviation (SCAD), Fan and Li (2001). It is known that the convex L_q penalty with $q > 1$ fails to satisfy the sparsity condition, while the convex L_1 penalty fails to satisfy the unbiasedness condition, and the concave L_q penalty with $0 \leq q < 1$ fails to satisfy the continuity condition. With these results, none of the L_q penalties is able to satisfy all three properties simultaneously. For this reason, Fan and Li (2001) introduced the SCAD which satisfies the three aforementioned properties. The SCAD penalty is given as

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

Elastic net, Zou and Hastie (2005). This variable selection method is a linear combination of L_1 and L_2 penalties. The main idea is to solve the optimization problem given as $\widehat{\beta} = argmin_{\beta}|y - X\beta|^2$, subject to $(1-\alpha)|\beta|_1 + \alpha|\beta|^2 \leq t$ for some t, and $\alpha = \frac{\lambda_2}{\lambda_1+\lambda_2}$. The function $(1-\alpha)|\beta|_1 + \alpha|\beta|^2$ is the elastic net penalty, which is a convex combination of the LASSO and ridge penalty. When $\alpha = 1$ and when $\alpha = 0$, the procedure becomes ridge regression and LASSO, respectively. One characteristic of the elastic net is its ability of selecting "grouped" variables, where strongly correlated predictors tend to be in or out of the model together. Moreover, the authors claim that this procedure in real data analysis and simulation studies outperform the LASSO in terms of prediction accuracy.

Dantzig selector, Candes and Tao (2007). This procedure is also based on penalized pseudo-likelihood which is a solution to the L_1 regularization problem. The idea is rather than controlling the size of the residuals, the Dantzig selector is based on minimizing $\|\beta\|_1$ subject to controlling the covariance vector $\|n^{-1}X^T(y - X\beta)\|_\infty \leq \lambda$, where $\lambda \geq 0$ is a regularization parameter. Dantzig selector's consistency for estimation and model selection depend heavily on the choice of λ. Shortly after the work on the Dantzig selector, it was observed that the Dantzig selector and the LASSO share some similarities.

Nevertheless, these methods are limited in handling ultrahigh dimensional problems due to the "curse of dimensionality". They are simultaneously challenged in terms of computational expediency, statistical accuracy and algorithmic stability (Fan et al. (2011)). Motivated by these concerns, Fan and Lv (2008) and Fan et al. (2009), developed a method that is based on correlation learning.

Correlation-based Marginal Methods

Sure Independence Screening (SIS), Fan and Lv (2008) and Fan et al. (2009). The main idea of SIS is to apply a two-stage procedure which involves screening out variables that have weak correlation with the response variable, and then applying lower-dimensional techniques such as SCAD, Dantzig selector and LASSO to further reduce the number of predictors and to estimate relevant parameters. SIS satisfies the sure screening property, that is, all the important variables are selected with probability tending to one under some conditions. In the screening stage, SIS ranks the independent variables in terms of their marginal correlations with the response. A problem in this stage is its failure to look at the joint correlation of the covariates to the response. Fan and Lv (2008) noted the possibility of the following problems in the SIS procedure: First, SIS may fail to select an important predictor that is jointly correlated but marginally uncorrelated or weakly correlated with the response; and second, when high collinearity exists among the predictors, SIS may select the unimportant predictors and exclude important predictors that are weakly correlated to the response.

Iterative Sure Independence Screening (ISIS), Fan and Lv (2008). To solve the problems in sure independence screening (SIS), the authors proposed the iterative-SIS (ISIS) which is an extension of SIS. The ISIS procedure works in the following manner: First, apply SIS and denote the set of selected variables as A_1 which contains k_1 variables. Obtain the residuals from regressing the response with the selected k_1 variables. In the next step, apply SIS procedure again with the residuals as the new responses and select k_2 variables from the $p - k_1$ variables. Denote this set as A_2 . Continue the iteration until there are l disjoint subsets $A_1, A_2, \ldots, A_l$ whose union has a size $d < n$. After the variable selection, apply a lower-dimensional techniques such as SCAD, Dantzig selector and LASSO to further reduce the number of predictors and to estimate relevant parameters. Despite of the sure screening properties of SIS-based procedures, they have methodological challenges: SIS relies on correlation which assumes that the response variable and covariates are normally distributed and the marginal relationship of each covariate with the response is linear. Hence, its assumptions are violated when the response variable is not normally distributed and when marginal or joint relationship of the predictors with the response is highly nonlinear.

Tilting method, Cho and Fryzlewicz (2012). The main task of this method is to select important variables in linear regression models. Unlike the SIS-based procedures, during variable selection, it takes into account both the marginal and joint relationship of the predictors with the response variable. To compute the correlation between a predictor, X_j and the response, X_j is first tilted. Tilting refers to transforming X_j by projecting it on the space orthogonal to the other predictors. This approach reduces the effect of other predictors on the tilted correlation of X_j and response Y.

When the joint distribution of the response variable and covariates does not follow a normal distribution, parametric methods based on conventional correlation may not be able to detect the true relationship between the response and the covariates and therefore, may lead to incorrect selection of covariates. In addition, the presence of nonlinear marginal or nonlinear joint relationship of the covariates with the response variable results in modeling biases in the linear model. To address these issues, there is a need for nonparametric variable selection procedures.

2.2.2 Variable Selection for Nonparametric Models

Nonparametric variable selection procedures can greatly improve model building and response estimation when parametric methods are not appropriate for the data. Fan et al. (2011) in their article said that using a nonparametric modeling procedure such as the additive model by Stone (1985) can significantly improve the flexibility of the ordinary linear model and allows transformed predictors to enter into the linear model. The additive model is given as $\mathbf{Y} = \sum_{j=1}^{p} m_j(X_j) + \varepsilon$. In this section, the limited number of procedures for variable selection in nonparametric additive models are discussed.

Penalized Methods

Penalized methods are popular approaches in variable selection and response estimation. The following are the currently available penalized methods for the nonparametric setting.

Extensions of LASSO. Lin and Zhang (2006) proposed a penalized procedure called Component Selection and Smoothing Operator (COSSO). This procedure is a functional generalization of LASSO using the Sobolev norm penalty, and it is able to carry out model selection on either additive or non-additive models. Another method is the penalized method for additive model (penGAM) by Meier et al. (2009). In this method, the authors combine the empirical L_2-norm and the usual roughness norm to enforce both sparsity and smoothness. The penGAM algorithm was built on the idea of a group LASSO problem. Ravikumar et al. (2009) also proposed Sparse additive models (SpAM) which is another generalization of the LASSO that uses the empirical L_2-norm of each additive component function. The method developed by Huang et al. (2010) is also an extension of LASSO to additive models.

Multiple Kernel Learning, Koltchinskii and Yuan (2010). This procedure is developed by combining empirical L_2-norms and kernel Hilbert space (RKHS norms). L_2 norms are used to impose sparsity of the final model while RKHS norms are to impose the smoothness of the components of the additive model.

Correlation-based Methods

The penalized methods applied to additive modeling are all challenged in terms of statistical accuracy, algorithmic stability and computational speed. Motivated by these concerns, methods based on correlation learning are developed.

Generalized Correlation for Feature Ranking (gcorr), Hall and Miller (2009).

The gcorr procedure is based on marginal screening procedure in which the generalized empirical correlations between the response and covariates are ranked. The authority of the ranks are assessed using bootstrap methods. The main goal of this procedure is to reduce the number of variables, after which other low-dimensional techniques such as LASSO may be applied for the prediction of the response. The gcorr procedure recruits a variable based on its marginal generalized correlation with the response Y and therefore, it ignores the collinearity that may exist among the independent variables. For this reason, gcorr leads to high false selection rate when the covariates are highly correlated.

Nonparametric Independence Screening (NIS) and its extensions, Fan et al. (2011). Nonparametric independence screening (NIS) is a nonparametric version of the sure independence screening (SIS). This procedure ranks the magnitude of marginal estimators, nonparametric marginal correlations and the marginal residual sum of squares. Specifically, the procedure fits the marginal regressions of each of the covariates with the response by employing a B-spline bases and then ranks their importance to the joint model based on the magnitude of the correlation of the marginal nonparametric estimate with the response. To select a set of variables, a threshold value is predefined. Aside from the marginal correlation, another equivalent approach of evaluating the importance of each covariate is by ranking the residual sum of squares of the componentwise nonparametric regressions. The authors extended the NIS procedure such as iterative NIS (INIS) and greedy-INIS, to reduce the false positive rate and stabilize the computation. After applying a NIS-based procedure for variable selection, a lower-dimensional technique is still required to further reduce the number of predictors and to estimate relevant parameters. NIS-based procedures have the sure screening property. However, NIS-based procedures rely greatly on marginal correlations and therefore, have high false selection rates when the covariates are highly correlated.

2.2.3 Generalized Additive Models, Continuous Case

Due to the curse of dimensionality for ultrahigh dimensional problems, many nonparametric methods fail to perform well. To avoid this problem, additive models was proposed by Stone (1985) which estimates the response Y using an additive approximation. Hastie and Tibshirani (1990) developed the generalized additive models (GAM) by extending additive models to a wide range of distribution families. GAM uses three techniques, namely, nonparametric regression, smoothing techniques and generalized distributional modeling. It can be applied when the relationship between the covariate and response is nonlinear, and when the distribution of the response variable belongs to the exponential family. The generalized additive model is defined as

$$Y = \alpha + \sum_{i=1}^{p} h_i(X_i) + \epsilon,$$

where $\{X_i\}$'s and ϵ are orthogonal, $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$. The function h_i's are smooth functions which are estimated in a nonparametric fashion. For univariate smoothing components, GAM procedure applies the B-spline and local regression methods, and the thin-plate smoothing spline for bivariate smoothing components. GAM uses the generalized cross validation (GCV) function as a criterion in choosing the smoothing parameters. The GCV function approximates the expected prediction error, and selects the model that has the smallest prediction error. As an alternative to using the GCV function, GAM also provides the option of specifying the degrees of freedom for each individual smoothing component.

2.3 Binary Case

2.3.1 Classification Techniques

Supervised learning with a qualitative response is considered as a classification problem. A classification problem can be further categorized into either binary classification or multi-class classification. The focus of this dissertation is binary classification. Classification is a technique that is used in many fields such as in bioinformatics, document classification and image recognition. One important area in bioinformatics is disease classification given ultrahigh dimensional datasets such as gene expressions and microarrays. Classification techniques have the goal to determine a function that can be used to predict the class in which a subject belongs given the independent variables or features. When the number of independent variables is much larger than the sample size, complications occur in most of classification procedures. Among the popular classification methods include logistic regression, Fisher's Discriminant analysis (FDA), support vector machines (SVM) and k-nearest neighbor classifier.

The response variable Y_i in classification problems is qualitative. It can assume two values for binary case or more than two values for the multiclass case. For example, in cancer classification, each of the covariates X_i represents the gene expression level of a patient and Y_i indicates whether this patient has cancer or not. Given a new observation X, classification techniques aim to predict the unknown class label Y of this new observation. In this section, various classification techniques are discussed.

Classical Methods. Many classical methods were developed that can be used for classification. Among the classical methods are Fisher's linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), and logistic regression. Bickel and Levina (2004) conducted a study to evaluate the performance of LDA in ultrahigh dimensional setting. In their results, LDA performs asymptotically no better than random guessing when the dimensionality p is much larger than the sample size n. For datasets with low dimension, the aforementioned classical methods perform very well. However, these methods fail when the number of covariates is much larger than the number of observations.

Distance-based classifiers. Various distance-based classifiers have been developed to deal with classification problems in ultrahigh dimensional setting. They tend to reduce the problems arising from the curse of dimensionality. Among the list of distance-based classifiers are naive-Bayes classifier, centroid rule, and k nearest neighbor rule.

The Bayes classifier conducts classification based on the posterior probabilities of the response. It is based on Bayes theorem and uses the following equation

$$P(Y = k|X = x) = \frac{P(X = x|Y = k)\pi_k}{\sum_{i=1}^{K} P(X = x|Y = i)\pi_i}. \tag{2.3.1}$$

The denominator of the equation does not need to be estimated even though it is unknown because it assumes a constant value. However, Bayes classifier breaks down in high dimensional settings due to curse of dimensionality and noise accumulation when estimating $P(X|Y)$. The naive-Bayes classifier overcomes this problem by assuming conditional independence which dramatically decreases the number of parameters to be estimated when modeling $P(X|Y)$. Specifically, the naive Bayes classifier uses the following equation:

$$P(X = x|Y = k) = \prod_{j=1}^{p} P(X_j = x_j|Y = k) \tag{2.3.2}$$

where X_j and x_j are the j^{th} components of X and x, respectively (see Fan et al. (2009) and the references therein). Hence, the conditional joint distribution of the p covariates depends only on marginal distributions. With this, naive-Bayes classifier is able to solve the problem of the curse of dimensionality. However, it assumes that the covariates are conditionally independent from each other even though they are not.

Another distance-based classifier is centroid classifier. It is a classification procedure which assigns a new observation to a class if its centroid is closest to the observation. The centroid could be the mean or median of data in class k. An extended version of centroid classifier has found applications in the medical domain, specifically classification of tumors (Tibshirani et al. (2002)).

The nearest neighbor classifier is another distance-based classifier which classifies new observations based on their similarity with observations in the training set. Given a new observation, the procedure finds the k closest observations in the training data set and assigns to the class that appears most frequently within the k-subset. To determine the k closest observations in the training data set, Euclidean distance is usually used. Larger k values may reduce the effects of noisy points within the training data set, and selecting the value for k is often done via cross-validation (Hall et al. (2005)).

Prediction Analysis for Microarrays (PAM). This approach was developed by Tibshirani et al. (2002) intended for cancer class prediction from gene expression profiling. This method is based on an improved version of the simple nearest centroid classier. Briefly, the method classifies a new observation based on shrunken standardized centroid for each class. Standardized centroid as explained by Tibshirani et al. (2002) in their paper is the average gene expression for each gene in each class divided by the within-class standard deviation for that gene. Nearest centroid classifiers obtains gene expression profile of a new observation, and compares it to the centroids of each class. The class with the nearest centroid, in squared distance, is the predicted class for that new observation. PAM uses the nearest shrunken centroid classifier in which each of the class centroids are shrunken toward the overall centroid for all classes by an amount called threshold. This shrinkage moves the centroid towards zero by threshold, setting it equal to zero if it hits zero. Nearest centroid classifier is then implemented to the shrunken class centroids. This shrinkage procedure in PAM has two advantages. First, it makes the classifier more accurate by reducing the effect of noisy genes; and second, it does automatic gene selection, that is, when a gene is shrunken to zero for all classes, then it is removed from the prediction rule. In addition, a special case of PAM sets a gene to zero for all classes except one, and high or low expression for that gene characterizes which class the new observations belong. To select the value for threshold, PAM does K-fold cross-validation for a range of threshold values. Typically, the threshold value chosen is the one which gives the minimum cross-validated misclassification error rate. Tibshirani et al. (2002) demonstrated PAM's effectiveness in finding genes for classifying small round blue cell tumors and leukemias.

Support Vector Machines (SVM). A support vector machine (SVM) was developed with the intention to classify new observations into two classes (Cortes and Vapnik (1995)). However, SVM may also be applied to multi-class problems by treating each single class as a separate problem. Given a training data, each marked as belonging to one of two categories, a model is obtained from SVM training algorithm. An SVM model is a representation of the observations in space, mapped so that the observations of the separate classes are divided by a hyperplane that is as wide as possible. The best hyperplane is the one that represents the largest separation between the two classes. New observations are then mapped into that same space and predicted to belong to a class based on which side of the hyperplane they lie on. SVM can perform both linear and nonlinear classification. Support vector machines are usually used in bioinformatics, image recognition and text categorization. It is shown in Dumais et al. (1998) that SVM outperforms other popular methods in text categorization, such as naive Bayes and decision trees in terms of prediction accuracy and computation time.

Chang and Lin (2011) have been actively developing a library for Support Vector Machines (LIBSVM). For classification, they developed c-Support Vector Classification (C-SVC) and v-Support Vector Classification (V-SVC). Given training variables $X_i \epsilon \mathcal{R}^n$, $i = 1, ..., l$ in two classes and a response binary variable Y, (C-SVC) solves the following problem:

$$\begin{gathered} \min_\alpha \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{subject to } y^T\alpha = 0,\ 0 \le \alpha_i \le C,\ i = 1, ..., l \end{gathered} \tag{2.3.3}$$

where $e = [1, ..., 1]^T$, Q is an $l \times l$ positive definite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is the kernel function. On the other hand, V-SVC introduces a new parameter $\nu \epsilon (0, 1]$ and the problem it solves is given as

$$\begin{gathered} \min_\alpha \frac{1}{2}\alpha^T Q\alpha \\ \text{subject to } 0 \le \alpha_i \le 1/l,\ i = 1, ..., l, \\ e^T\alpha \ge \nu,\ y^T\alpha = 0 \end{gathered} \tag{2.3.4}$$

where $Q_{ij} \equiv y_i y_j K(x_i, x_j)$.

Classification Tree Based Methods. Decision tree methods according to Rokach and Maimon (2008) are commonly used in data mining. The main goal is to build a model that predicts the value of a response variable based on several independent variables. Tree based classification methods was first introduced by Breiman et al. (1984). Three of the popular tree based procedures are bagging, random forest and boosted trees. Bagging decision trees as discussed by Breiman (1996) builds numerous decision trees by conducting a multiple resampling of training data with replacement and classification of a new observation is based on majority vote among the trees.

On the other hand, random forest was introduced independently by Ho (1995) and Amit and Geman (1997). It is a combination of "bagging method" and random selection of features. To construct a model using random forest requires making choices for the shape of the decision to use for every node, the type of predictor to use for every leaf, the splitting objective to optimize for every node and the method for implementing randomness into the trees. For classification, the new observation is entered to the tree and is assigned to a class corresponding to the node where it ends up. This procedure is iterated over all trees and the observation is classified based on the majority vote of the trees.

Another tree-based method is "boosting" which iteratively grows classification trees in a sequence of reweighted datasets (Austin and Lee (2011)). In a given iteration, subjects who were misclassified in the previous iteration are given higher weights than those who were correctly classified. The final classification is based on the majority vote of classification trees.

2.3.2 Variable Selection Techniques

Variable selection is a procedure to determine the covariates that have strong contribution to the response variable. It is very important to conduct variable selection in ultrahigh dimensional setting to avoid the curse of dimensionality. Most classification techniques fail when the number of covariates is much larger than the number of observations. Hence, for efficient classification of qualitative response variables, it is necessary to first reduce the number of covariates before implementation of classification procedures. Majority of the variable selection procedures were already discussed in Sections 2.2 and 2.3. These procedures can also be used for variable selection when the response variable is binary. In addition to these procedures, this section presents the popular variable selection procedures that are specifically used for variable selection when the response variable is binary, namely, Shrinkage methods, SVM, GLMNET, Gene Selection with Random Forest (GeneSrF) and Binary Matrix Shuffling Filter (BMSF).

Shrinkage methods. The shrinkage methods were developed for regression problems with the objective of shrinking the coefficients of some variables by imposing a penalty on their size. Among the shrinkage methods are Ridge regression and LASSO which were already discussed in Section 2.2.

Variable Selection in SVM. Guyon et al. (2002) developed a variable selection procedure called RFE-SVM which stands for recursive feature elimination using binary support vector machine. This procedure ranks genes using the coefficient magnitude trained from the SVM instead of ranking genes using correlation between gene and phenotype. It recursively removed covariates with smallest coefficient magnitude in the learned SVM model followed by recursively training the updated data with decreasing number of variables to re-rank the rest of genes. The authors demonstrated that the RFE-SVM has the ability to eliminate gene redundancy automatically and yielded better and more compact gene subsets. Moreover, Fan and Li (2001) also developed SCAD-SVM. Both RFE-SVM and SCAD-SVM were developed using binary SVM.

Variable Selection in Random Forest. A very popular random forest procedure that can be used for binary variable selection is the Gene Selection in Random Forest (GeneSrF) which was developed by Diaz-Uriarte and Alvarez de Andres (2006) to select relevant genes to be used for classification in gene expression studies. Its main goal is to identify the smallest possible number of genes and still result in good predictive performance. The authors have demonstrated the performance of GeneSrF through simulated and microarray data sets. In their results, random forest has comparable predictive performance to other classification methods, including KNN and SVM. In addition, GeneSrF in most data sets have yielded smaller sets of genes than alternative methods while preserving predictive accuracy.

Binary Matrix Shuffling Filter (BMSF). Zhang et al. (2012) developed the Binary Matrix Shuffling Filter (BMSF) for variable selection. This method takes into account possible gene interactions during gene selection. To perform filtering, BMSF utilizes Support Vector Machine (SVM) and eliminate variables through evaluating the effect of random sets of genes. During the gene selection process, the set of genes kept in the model was repeatedly refined and updated while taking into account the effect of a given gene on the contributions of other genes to their importance in cancer classification. Through real data sets, the authors have shown that BMSF often selects very small number of genes while preserving predictive accuracy. The significance of using BMSF includes: (1) It accounts for possible gene interactions, (2) It often selects small number of genes while accurately classify new observations, (3) It results in improved LOOCV classification accuracy when coupled with SVM, Naive Bayes (NB), linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). Results from Zhang et al. (2012) suggests that accounting for interactions among features in the search space coupled with a manageable search scheme as in BMSF provides better accuracy for biomarker selection.

Generalized Linear Models with Elastic Net (GLMNET). This procedure was developed by Jerome Friedman, Trevor Hastie and Rob Tibshirani which contains very efficient procedures for fitting elastic-net regularization paths for generalized linear models. The elastic net penalty includes mixture of ridge and lasso penalties. The GLMNET function can fit Gaussian and multiresponse Gaussian models, logistic regression, poisson regression, multinomial and grouped multinomial models and the Cox proportional hazard model. The efficiency of the GLMNET algorithm comes from using cyclical coordinate descent in the optimization process and from underlying Fortran code. The coordinate descent update has the form

$$\widetilde{\beta}_j \leftarrow \frac{S(\frac{1}{N}\sum_{i=1}^{N} w_i x_{ij}(y_i - \widetilde{y}_i^{(j)}), \lambda\alpha)}{\sum_{i=1}^{N} w_i x_{ij}^2 + \lambda(1-\alpha)} \tag{2.3.5}$$

where $\widetilde{y}_i^{(j)} = \widetilde{\beta}_0 + \sum_{l\neq j} x_{il}\widetilde{\beta}_l$ pertains to the fitted value excluding the contribution from x_{ij}. The $S(z,\tau)$ is the soft thresholding operator defined as

$$sign(z)(|z|-\tau)_+ = \begin{cases} z-\tau & \text{if } z>0 \text{ and } \tau<|z| \\ z+\tau & \text{if } z<0 \text{ and } \tau<|z| \\ 0 & \text{if } \tau \geq |z| \end{cases}$$

For detailed discussion of GLMNET, please refer to Friedman et al. (2009).

2.3.3 Generalized Additive Model (GAM), Binary Case

Wood (2008) discussed the generalized additive model when the response variable is binary, that is when the outcome y_i is either 0 or 1. The value 1 indicates an event and 0 indicates no event. The objective for GAM-binary case is to model $p(y|X)$ which is defined as the probability of an event given $X = (x_1, x_2, ...x_p)^T$. The generalized additive logistic model assumes that

$$\begin{aligned} logit(p(Y=1|X)) &= log\frac{p(y|X)}{1-p(y|X)} \\ &= f_0 + \sum_{j=1}^{p} f_j(x_{ij}) \\ &= \eta(x) \end{aligned} \tag{2.3.7}$$

where the f_j's, $j = 1, ..., p$ are smooth functions obtained via thin-plate smoothing. The probability of an event given $X = (x_1, x_2, ...x_p)^T$ is

$$p(Y=1|X) = \frac{e^{(f_0+\sum_{j=1}^{p} f_j(x_{ij}))}}{1+e^{(f0+\sum_{j=1}^{p} f_j(x_{ij}))}} = \frac{e^{(\widehat{\eta})}}{1+e^{(\widehat{\eta})}}. \tag{2.3.8}$$

GAM uses the generalized cross validation (GCV) function as a criterion in choosing the smoothing parameters. The GCV function approximates the expected prediction error, and selects the model that has the smallest prediction error. As an alternative to using the GCV function, GAM also provides the option of specifying the degrees of freedom for each individual smoothing component.

Chapter 3

Continuous Case: Variable Selection and Prediction

3.1 Introduction

In ultrahigh dimensional settings, high collinearity among the covariates is likely to exist, which makes marginal correlation screening unreliable as a measure of association between the variables and the response. Specifically, the existing nonparametric procedures are challenged by the following problems (Fan and Lv (2008)):

1. Unimportant covariates are likely to enter the final model when they are highly correlated with important covariates.

2. Important covariates that are marginally uncorrelated but jointly correlated with the response are unlikely to enter the final model.

3. There exists a problem of collinearity among the covariates.

Given these problems, nonparametric marginal screening procedures such as the NIS-based procedures have high false selection rates in the final model. Hence, this paper proposes the Most Significant Regression - Continuous (MSRc) algorithm which is a variable selection and response estimation procedure that takes into account the correlation structure among the covariates. The algorithm is a combination of smoothing spline estimation, additive modeling and tests of generalized conditional correlation procedures. It can be used with continuous or discrete response variables, and when the predictors are linearly or nonlinearly related to the response.

Comparisons with other methods such as NIS, INIS, greedy-INIS and generalized correlation (gcorr) are presented using the results from Monte Carlo simulations. Using a real data from genome wide association studies (GWAS), the prediction accuracy of MSRc is also compared with Support Vector Regression (SVR) Models and Bayesian LASSO which are established feature selection procedures in GWAS.

3.2 GAM in the Continuous Case

Generalized Additive Modeling (GAM) technique by Wood (2008) is implemented in the MSRc algorithm and therefore it is important to present how it was used. GAM is used to assess the significance of a smoothing spline estimate of X, say $f(x)$, in predicting the response, say Y. The notation used in this paper is $GAM(Y, f(x))$. Given the variables Y and X, GAM derives the smoothing spline estimate $f(x)$ that minimizes the function

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int_{-\infty}^{\infty}[f''(x)]^2 dx.$$

The term $\int_{-\infty}^{\infty} f''(x)^2 dx$ measures how wiggly $f(x)$ is and $\lambda \geq 0$ is how much $f(x)$ is penalized for being wiggly. In this paper, a thin plate regression spline basis was used and the value of λ was selected to yield an effective degrees of freedom which is controlled by the degree of penalization selected during fitting by generalized cross validation (GCV) criterion given as

$$nD/(n - DoF)^2,$$

where D refers to deviance of the model computed as $D = \sum_{i=1}^{n}(y_i - \widehat{f}(x_i))^2$, and $\widehat{f}(x_i)$ is the estimate from fitting to all the data. DoF is the effective degrees of freedom of the model and n is the number of observations (Wood (2008)).

3.3 Most-Significant-Regression Algorithm, MSRc

Suppose that we have a random sample $(\mathbf{X}_i, Y_i)$, $i = 1, 2, ..., n$ observed from an unknown population, where $\mathbf{X}_i = (X_{i1}, \ldots, X_{ip})^T$. We consider the case that $p >> n$. Suppose that only a small subset of covariates of size p' contribute to the response and $p' < p$. For convenience of notation, we denote these p' covariates as $\mathbf{Z}_i = (X_{ij_1}, \ldots, X_{ij_{p'}})^T$. Moreover, let the true model be

$$\mathbf{Y}_i = m(\mathbf{Z}_i) + \varepsilon_i \tag{3.3.1}$$

in which Y_i is continuous and ε_i is the random error with conditional mean equal to 0. The covariates in $\mathbf{Z}_i$ are called active variables which we want to identify from the entire set of covariates $\Omega = \{X_{i1}, \ldots, X_{ip}\}$. Let $\mathbf{X}^* = \{X_1^*, ..., X_m^*\}$ be the set of variables selected to enter the model. Initialize $\mathbf{X}^* = \phi$.

Step 1: Let $k = 1$, $\varepsilon_0(\mathbf{X}^*) = Y$, and $\widehat{g} = 0$. The $\widehat{g}$ stores the fitted values.
Step 2: Select the first variable marginally as follows. Calculate p-values of test $GAM(\varepsilon_0(\mathbf{X}^*), \widehat{f}(X_j))$, for $j = 1, \ldots, p$, where $\widehat{f}(X_j)$ is a smoothing spline estimate of $\varepsilon_0(\mathbf{X}^*)$, $j = 1, \ldots p$. The p-values are obtained for testing individual smooth terms for equality to the zero function. Choose the variable in Ω that has the smallest significant p-value (pvalue$< \alpha = 0.01$) and denote it as X_k^*. If X_k^* does not exist, terminate the algorithm. Otherwise, proceed to Step3.
Step 3: Fit $\varepsilon_{k-1}(\mathbf{X}^*)$ with the smoothing spline estimate of X_k^*, $\widehat{f}(X_k^*)$. Update $\mathbf{X}^* \Leftarrow \mathbf{X}^* \bigcup \{X_k^*\}$ and $\mathbf{X_{new}} \Leftarrow \Omega - \mathbf{X}^*$.
Step 4: Update $\widehat{g} \Leftarrow \widehat{g} + \widehat{f}(X_k^*)$ and compute $\varepsilon_k(\mathbf{X}^*) = Y - \widehat{g}$. If $\mathbf{X_{new}} = \phi$, terminate the algorithm.
Step 5: Calculate p-value of test $GAM(Y, \widehat{g})$. Denote it as p_c.
Step 6: Calculate p-values of test $GAM(\varepsilon_k(\mathbf{X}^*), \widehat{f}(X_j))$ for all $X_j \epsilon \mathbf{X_{new}}$. If there is no p-value close enough to p_c, that is, $|pvalue - p_c| < 0.0005$ terminate the algorithm. Other-

wise, proceed to the next step.

Step 7: Set $k = k+1$. Select the independent variable whose p-value is close enough to p_c, that is, $|pvalue - p_c| < 0.0005$ and denote the variable as X_k^*. Then return to Step3.

The algorithm uses $|pvalue - p_c| < 0.0005$ as a criterion for recruiting an independent variable. The effects of this criterion are different for two scenarios.

1. If the first variable is good, that is, p_c is small (close to zero), then the selection ensures that the next variable recruited is highly significant.

2. The first variable recruited may be a false positive when active variables are marginally uncorrelated with Y, or if inactive variables are highly correlated with some active variables. When the first variable recruited is a false positive, then the algorithm controls the contribution of the next variable being selected (which may also be a false positive) to be no more than the first variable in the model. The contribution is in terms of generalized correlation to be explained below.

In the algorithm, p-values from $GAM(\varepsilon(\mathbf{X}^*), \widehat{f}(X_j))$ for all $X_j \, \epsilon \, \mathbf{X_{new}}$ test the significance of the generalized correlation between the current residual $\varepsilon(\mathbf{X}^*)$ and each of the smoothed function of the remaining $X_j \in \mathbf{X_{new}}$. Since the residuals are calculated based on the variables that are already in the model, this correlation is actually the conditional generalized correlation between Y and $X_j \in \mathbf{X_{new}}$ conditional on the variables selected in earlier steps. The generalized correlation of $\varepsilon(\mathbf{X}^*)$ and $f(X_j)$ is given and estimated by

$$\vartheta_j = sup_{f \epsilon \mathfrak{F}} \frac{cov(f(X_j), \varepsilon(X^*))}{\sqrt{var(f(X_j))}\sqrt{var(\varepsilon(X^*))}} \tag{3.3.2}$$

$$\widehat{\vartheta}_j = sup_{f \epsilon \mathfrak{F}} \frac{\sum_i (f(X_{ij}) - \overline{f_j})(\varepsilon_i(X^*) - \overline{\varepsilon}^*)}{\sqrt{n \sum_i (f(X_{ij})^2 - \overline{f}_j^{\,2})}\sqrt{n \sum_i (\varepsilon_i(X^*)^2 - \overline{\varepsilon}^{*2})}} \tag{3.3.3}$$

respectively, where $\overline{f_j} = n^{-1} \sum_i f(X_{ij})$ and $\overline{\varepsilon}^* = n^{-1} \sum_i \varepsilon_i(X^*)$.

3.3.1 Comparison of Greedy INIS and MSRc

Greedy-INIS ranks the utility of covariates according to a measure of goodness of fit such as the magnitude of marginal estimators, nonparametric marginal correlations, and the marginal residual sum of squares. In the variable screening, via thresholding, it selects a set of variables with size less than or equal to p_0, a small positive integer. An example of the threshold value is the q^{th} quantile of the marginal correlations. There are three problems involved with this thresholding step of greedy-INIS. First, what is the most reliable threshold value? Does the threshold value depend on the judgment of the person who runs the analysis? Second, given thousands of independent variables, the threshold value selected may result in hundreds of variables being recruited. This will lead to achieving sure screening property such that all the positives are selected but very high false selection rates. Some threshold value may also result in very few variables selected which leads to very low false selection rates but very low true positive rates. Third, since greedy-INIS relies mainly on the threshold value, it will always select at least one variable to enter even though there are no variables important in predicting the response. In the case of highly correlated independent variables and when there is only one important variable used to generate Y, greedy-INIS is likely to select at least 1 variable. After variable screening, the next step is to select the final variables to enter the model by applying a more refined technique such as penGAM (Meier et al. (2009)) and SCAD (Fan and Li (2001)). Greedy-INIS mainly rely on these refined techniques to improve its false selection rates.

When $p_0 = 1$, greedy-INIS and MSRc both recruit one variable at a time. The major difference is the criteria on how they recruit the variable. As stated above, greedy-INIS selects a variable based on a fixed threshold value. On the other hand, MSRc selects the variable through two criteria. The first criterion applies to the recruitment of the first covariate, and it requires that the smoothing spline estimate $\widehat{f}(X_1^*)$ of a variable X_1^* should be significant (p-value $< \alpha = 0.01$) in predicting the response. The second criterion specifies that X_k is recruited if the significance of its smoothing spline estimate $\widehat{f}(X_k)$ of the current residual is comparable to the significance of the current regression estimate of the response. The threshold value in MSRc pertains to p_c in Step 5 of the MSRc algorithm which is updated every time a new variable is being added to the model. As a new variable is being added to the model, the residuals $\varepsilon_k(\mathbf{X}^*)$ also change. The p-values of $GAM(\varepsilon_k(\mathbf{X}^*), \widehat{f}(X_i))$ for all $X_i \epsilon \mathbf{X_{new}}$ explained in Steps 6 and Step 7 of the MSRc algorithm also change as the residuals change. Hence, MSRc recruits a variable X_k based on the significance of $\widehat{f}(X_k)$ in predicting the current residual assessed relative to the contributions of the variables that are already in the model.

3.4 Performance Measures

To compare the performance of MSRc with existing nonparametric procedures, this paper uses the mean true positive rate (TP), mean false positive rate (FP) and mean squared prediction error (PE). Specifically, for variable selection properties, the following measures are computed:

$$TP = \frac{\sum_{i=1}^{r} \text{number of active variables recruited in the } i^{th} \text{ run}}{r} \quad ,$$

$$FP = \frac{\sum_{i=1}^{r} \text{number of inactive variables recruited in the } i^{th} \text{ run}}{r} \quad ,$$

where r is the number of runs. Active variables are the independent variables used to generate the true model, while inactive variables are the independent variables not used in the true model. For predictive performance, independent testing is implemented to avoid overfitting that occurs when the variable selection, model building and testing are applied to the same data. That is, the prediction error (PE) is calculated on an independent test data set of size $n/2$. The PE is computed using the equation

$$PE = \frac{\sum_{i=1}^{r} \sum_{j=1}^{n/2} \frac{(\widehat{y}_{ij} - y_{ij})^2}{n/2}}{r} \quad .$$

3.5 Graphical Presentation of the MSRc Algorithm

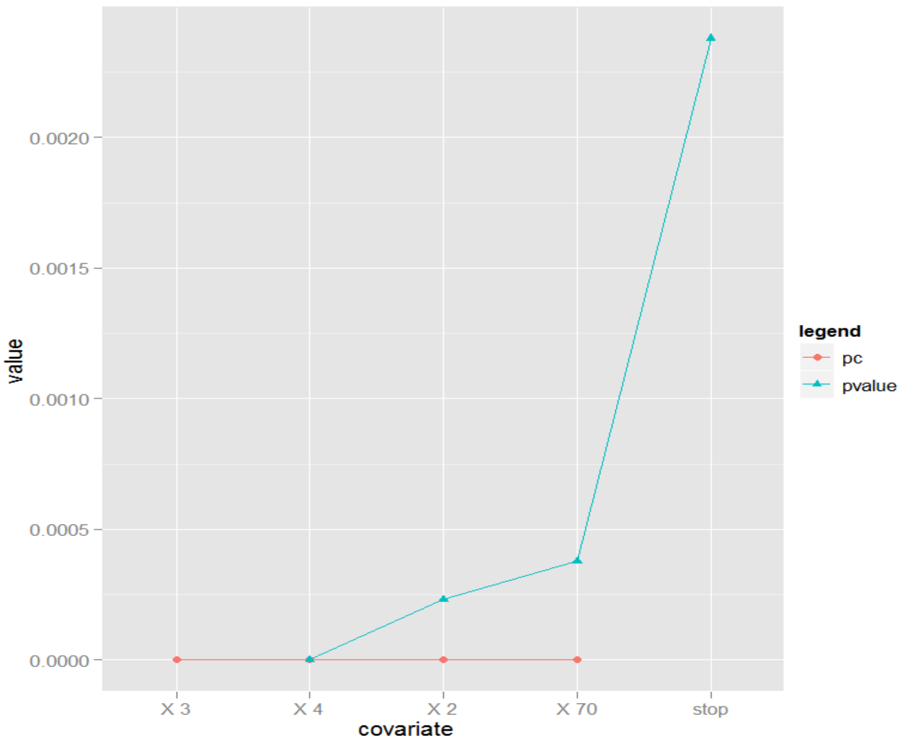
Here we illustrate the variable selection process of the MSRc algorithm with three examples. These examples use simulated data from models discussed in earlier articles.

Example 1. This is an example also discussed in Hall and Miller (2009). Suppose W_{ij}, $j = 1, \ldots, 6$ and X_{ik}, $k = 5, \ldots, 5000$ are independent random variables which follow a $N(0, 1)$ distribution, and let $Y_i = 2sin\{\frac{\pi}{2}(W_{i1} + 0.5W_{i2})\} + \sum_{j=3}{}^{5} W_{ij}{}^{2} + 0.4e^{W_{i6}} + Z_{i0}$, $X_{i1} = 2W_{i1}{}^{2} + Z_{i1}$, $X_{i2} = 2W_{i2} + Z_{i2}$, $X_{i3} = W_{i3}W_{i4} + Z_{i3}$, and $X_{i4} = W_{i6} + Z_{i4}$, where each of the Z_{ij}'s are normally distributed with mean equal to 0 and standard deviation of 0.1. The sample size is $n = 500$. This is a measurement error model in that the true covariates $W_{ij}, j = 1, \ldots, 6$ are not directly observed.

Figure 3.1 presents MSRc algorithm for one simulation of data from Example 1. The first variable recruited is X_3 which has a p-value equal to 1.69×10^{-38}. It is significant at $\alpha = 0.01$ and it is the smallest among all p-values from each of the 5000 generated covariates. At this point, p_c is set to be equal to 1.69×10^{-38}. The second variable recruited is X_4 which has p-value equal to 1.59×10^{-25}. Note that the absolute difference of the current p_c and p-value of X_4 is less than 0.0005. After X_4 is recruited, p_c becomes 3.39×10^{-67}. Next, X_2 is recruited with p-value equal to 2.31×10^{-4}. Again, the absolute difference of the current p_c and p-value of X_2 is less than 0.0005. After recruiting X_2, the value of p_c becomes 5.11×10^{-72}. The variable recruited next is X_{70} with p-value equal to 3.77×10^{-4}. The absolute difference of the current p_c and p-value of X_{70} is less than 0.0005. The current p_c is now equal to 3.36×10^{-79}. The algorithm proceeds to finding the variable whose p-value is closest to the current p_c. The p-value closest to the current p_c is 2.39×10^{-3}. However, $|2.39 \times 10^{-3} - 3.36 \times 10^{-79}| = 0.002$ which is greater than 0.0005. Hence, the algorithm stops.

Result comparison. For this example, the generalized correlation (gcorr) procedure by Hall and Miller (2009) implemented 500 bootstrap simulations with size $n = 500$ and used a prediction level of $\alpha = 0.02$. With the lowest 99% percentile ranking plot, Hall and Miller (2009) selected 10 variables X_3, X_4, X_{3484}, X_{3010}, X_{2672}, X_{1264}, X_{3275}, X_{307}, X_{2787},

Figure 3.1: *MSRc algorithm for Example 1. The covariates were recruited in the following order: X3, X4, X2 and X70. The algorithm stopped when the* $|pvalue - p_c| \geq 0.0005$*. The p-value is the significance of* $f(x_k)$ *in predicting the current residual, while* p_c *is the significance of the current regression estimate of the response.*



	pc	pvalue
X 3	1.694325e-38	NA
X 4	3.386070e-67	1.589661e-25
X 2	5.113372e-72	2.307805e-04
X 70	3.355634e-79	3.774648e-04
stop	NA	2.378705e-03

and X_{459}. The X_3 and X_4 were clearly shown as more influential in their plot than the other 8 covariates. In summary, the Hall and Miller (2009) procedure identified two true positives but eight false positives. On the other hand, MSRc algorithm found three true positives X_2, X_3 and X_4, and only one false positive.

Example 2. This example generates data based on a model that was also considered in Meier et al. (2009) and Fan et al. (2011). Following Fan et al. (2011), we set $n = 400$ and $p = 1000$. Let

$$g_1(x) = x, \qquad g_2(x) = (2x-1)^2, \qquad g_3(x) = \frac{\sin(2\pi x)}{2 - \sin(2\pi x)}$$

and

$$g_4(x) = 0.1\sin(2\pi x) + 0.2\cos(2\pi x) + 0.3\sin^2(2\pi x) + 0.4\cos^3(2\pi x) + 0.5\sin^3(2\pi x).$$

The response variable was generated from the following additive model:

$$Y = 5g_1(X_1) + 3g_2(X_2) + 4g_3(X_3) + 6g_4(X_4) + \sqrt{1.74}\varepsilon$$

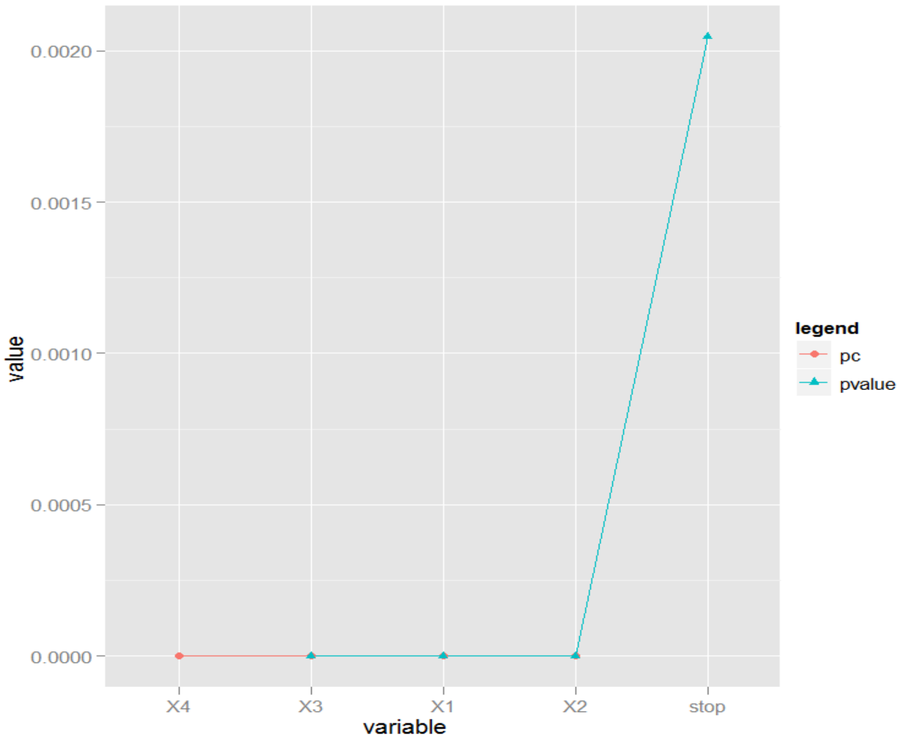
The independent variables $X = (X_1, \ldots, X_p)^T$ are simulated according to the random-effects model

$$X_j = \frac{W_j + tU}{1+t}, \qquad j = 1, \ldots, p,$$

where $W_1, \ldots, W_p$ and U are iid Unif(0,1), and $\varepsilon \sim$ N(0,1) independent of $\{X_i\}$'s. When $t = 0$, the covariates are all independent, and when $t = 1$, the pairwise correlation of covariates is 0.5.

Graphical Presentation of MSRc Algorithm. Figure 3.2 is the graphical presentation of MSRc algorithm for one simulation of data from Example 2 with $t = 0$. The first variable recruited is X_4 which has a p-value equal to 4.06×10^{-69}. It is significant at $\alpha = 0.01$ and it is the smallest among all p-values from the 1000 generated covariates. At this point, p_c is set to be equal to 4.06×10^{-69}. The second variable recruited is X_3 which has p-value equal to 2.05×10^{-39}. Note that the absolute difference of the current p_c and p-value of X_3 is less than 0.0005. After X_3 is recruited, p_c becomes 8.24×10^{-110}. The next variable recruited is X_1 with p-value equal to 6.30×10^{-49}. Again, note that the absolute difference of the current p_c and p-value of X_1 is less than 0.0005. After recruiting X_1, the value of p_c becomes 7.61×10^{-159}. The variable recruited next is X_2 with p-value equal to 2×10^{-26}. The absolute difference of the current p_c and p-value of X_2 is less than 0.0005. The current p_c is now equal to 2.56×10^{-190}. The algorithm proceeds to finding the variable whose p-value is closest to the current p_c. The p-value closest to the current p_c is 2.05×10^{-3}. However, $|2.05 \times 10^{-3} - 2.56 \times 10^{-190}| = 0.002$ which is greater than 0.0005. Hence, the algorithm stops.

Figure 3.2: *MSRc algorithm for Example 2 (t=0). The covariates were recruited in the following order: X_4, X_3, X_1 and X_2. The algorithm stopped when the $|pvalue - p_c| \geq 0.0005$. The p-value is the significance of $f(x_k)$ in predicting the current residual, while p_c is the significance of the current regression estimate of the response.*



	pc	pvalue
X4	4.058269e-69	NA
X3	8.237006e-110	2.047447e-39
X1	7.610119e-159	6.302985e-49
X2	2.556754e-190	1.999054e-26
stop	NA	2.047515e-03

The results of Meier et al. (2009) and Fan et al. (2011) were not reported for a single run. Instead, they reported a summary of the results from multiple runs. We defer such comparisons to a later section.

Example 3. Again, this simulation model is from the paper of Fan et al. (2011) which was first implemented by Meier et al. (2009). There are 12 active variables with different coefficients.

$$\begin{aligned} Y &= g_1(X_1) + g_2(X_2) + g_3(X_3) + g_4(X_4) + 1.5g_1(X_5) + 1.5g_2(X_6) \\ &\quad + 1.5g_3(X_7) + 1.5g_4(X_8) + 2g_1(X_9) + 2g_2(X_{10}) + 2g_3(X_{11}) + 2g_4(X_{12}) + \sqrt{0.5184}\varepsilon \end{aligned}$$

where $\varepsilon \sim$ N(0,1) independent of $\{X_i\}$'s. The covariates are simulated as in Example 2 and we also set $n = 400$ and $p = 1000$.

Graphical Presentation of MSRc Algorithm. Figure 3.3 is the graphical presentation of MSRc algorithm for one simulation of data from Example 3 (t=0). The first variable recruited is X_{12} which has a p-value equal to 2.40×10^{-24}. It is significant at $\alpha = 0.01$ and it is the smallest among all p-values from each of the 1000 covariates. At this point, p_c is equal to 2.40×10^{-24}. The second variable recruited is X_8 which has p-value equal to 7.77×10^{-14}. Note that the absolute difference of the current p_c and p-value of X_8 is less than 0.0005. After X_8 is recruited, p_c becomes 8.72×10^{-40}. The variables entered next in order are X_{11}, X_7, X_9, X_{10}, X_5, X_4, X_3, X_6, X_1 and X_2. After recruiting all of these 12 covariates, the algorithm proceeds to finding the next variable to recruit. At this point, the current p_c is equal to 6.85×10^{-181}. The p-value closest to the current p_c is 1.43×10^{-3}. However, $|1.43 \times 10^{-3} - 6.85 \times 10^{-181}| = 0.001$ which is greater than 0.0005. Hence, the algorithm stops.

We defer the numerical comparisons with the results of Meier et al. (2009) and Fan et al. (2011) in next section.

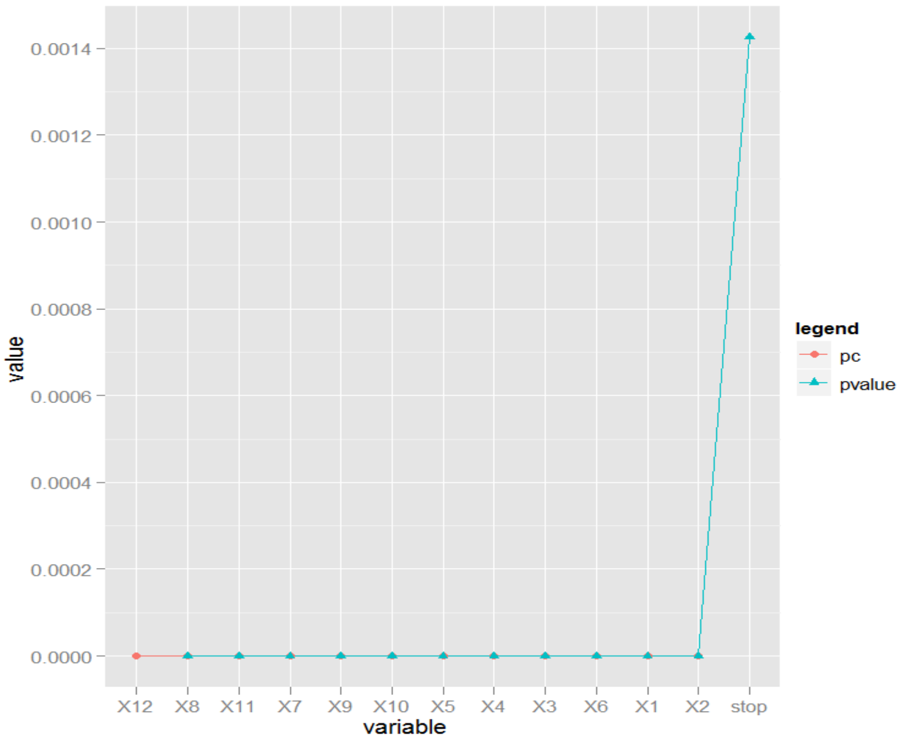
3.6 Numerical Comparisons

3.6.1 Simulation Models and Results

To demonstrate the power of our proposed MSRc method and to be able to compare with the results of Fan et al. (2011) and Hall and Miller (2009), we present comparison results for the following 8 simulation examples.

Example 2 continued. Similar to Fan et al. (2011), we generated data with the model in Example 2 and repeated the data generation 100 times. For each set of data generated, we applied our MSRc algorithm. The summary of the results from all 100 runs are given in Table 3.1.

Figure 3.3: *MSRc algorithm for Example 3 with t=0. The covariates were recruited in the following order:* X_{12}, X_8,X_{11}, X_7, X_9, X_{10}, X_5, X_4, X_3, X_6, X_1 *and* X_2. *The algorithm stopped when the* $|pvalue{-}p_c| \geq 0.0005$. *The p-value is the significance of* $f(x_k)$ *in predicting the current residual, while* p_c *is the significance of the current regression estimate of the response.*



	pc	pvalue
X12	2.396003e-24	NA
X8	8.715722e-40	7.774712e-14
X11	5.385543e-58	2.257844e-15
X7	1.346557e-74	8.107431e-12
X9	4.766031e-89	1.013635e-14
X10	1.469327e-101	1.296765e-10
X5	1.895683e-113	2.610391e-13
X4	5.622542e-128	3.646222e-11
X3	2.802539e-142	8.308468e-13
X6	7.245347e-161	2.749789e-15
X1	2.629366e-166	8.475396e-09
X2	6.846030e-181	9.601577e-10
stop	NA	1.426431e-03

Based on Table 3.1, the Most Significant Regression algorithm, MSRc, consistently has competitive true positive rates and has lowest false selection rates for both cases, that is, when covariates are independent ($t = 0$) and when pairwise correlation of covariates is 0.5 ($t = 1$). In terms of predictive performance, MSRc achieves the smallest prediction error for both cases.

Table 3.1: *Mean True Positive Rate and Mean False Positive Rate for Example 2. There are 100 simulations each of size 400, from which the mean true positive rate (TP) and mean false positive rate (FP) were computed. The prediction errors were computed from an independent test data of size 200 for each simulation. The mean prediction error (PE) was computed from the results of 100 simulations. Robust standard deviations are given in parentheses.*

Model	Method	TP	FP	PE
Example 2 t=0	MSRc	4.00(0.00)	0.43(0.75)	2.15(0.26)
	INIS	4.00(0.00)	2.58(2.24)	3.02(0.34)
	g-INIS	4.00(0.00)	0.67(0.75)	2.92(0.30)
	penGAM	4.00(0.00)	31.86(23.51)	3.30(0.40)
	ISIS	3.03(0.00)	29.97(0.00)	15.95(1.74)
Example 2 t=1	MSRc	3.99(0.00)	0.16(0.00)	2.29(0.24)
	INIS	3.98(0.00)	15.76(6.72)	2.97(0.39)
	g-INIS	4.00(0.00)	0.98(1.49)	2.61(0.26)
	penGAM	4.00(0.00)	39.21(24.63)	2.97(0.28)
	ISIS	3.01(0.00)	29.99(0.00)	12.91(1.39)

Example 3 continued. Here we report comparisons based on 100 runs. The mean true positive rate (TP), mean false positive rate (FP), and prediction error (PE) from independent test data sets are given in Table 3.2. Table 3.2 shows that when the covariates are independent ($t = 0$), the MSRc algorithm is competitive with other methods in terms of mean true positive rate. In comparison, MSRc has the lowest mean false positive rate and lowest prediction error. When the covariates have pairwise correlation equal to 0.5, that is, when ($t = 1$), MSRc has the lowest mean false positive rate. However, it is slightly worse than the other methods in terms of mean true positive rate and prediction error.

Example 4. This example is the simulation model used by Fan et al. (2009) and implemented by Fan et al. (2011) in their paper. We set $n = 400$ and $p = 1000$. The response Y

Table 3.2: *Mean True Positive Rate and Mean False Positive Rate for Example 3 from 100 runs. There are 100 simulations each of size 400, from which the mean true positive rate (TP) and mean false positive rate (FP) were computed. The prediction errors were computed from an independent test data of size 200 for each simulation. The mean prediction error (PE) was computed from the results of 100 runs. Robust standard deviations are given in parentheses.*

Model	Method	TP	FP	PE
Example 3 t=0	MSRc	11.98(0.00)	0.47(0.75)	0.88(0.12)
	INIS	11.97(0.00)	3.22(1.49)	0.97(0.11)
	g-INIS	12.00(0.00)	0.73(0.75)	0.91(0.10)
	penGAM	11.99(0.00)	80.10(18.28)	1.27(0.14)
	ISIS	7.96(0.75)	25.04(0.75)	4.70(0.40)
Example 3 t=1	MSRc	8.25(1.49)	0.70(0.75)	1.53(0.22)
	INIS	10.01(1.49)	15.56(0.93)	1.03(0.13)
	g-INIS	10.78(0.75)	1.08(1.49)	0.87(0.11)
	penGAM	10.51(0.75)	62.11(26.31)	1.13(0.12)
	ISIS	6.53(0.75)	26.47(0.75)	4.30(0.44)

is simulated from the model $Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \varepsilon$, where $\varepsilon \sim$ N(0,1). The independent variables $X_1, \ldots, X_p$ are jointly Gaussian, and each variable has a marginal distribution N(0,1), and $corr(X_i, X_4) = \frac{1}{\sqrt{2}}$ for all $i \neq 4$ and $corr(X_i, X_j) = \frac{1}{2}$ if i and j are distinct elements of $\{1, \ldots, p\} \setminus \{4\}$. The values of the coefficients are set to $\beta_1 = 2$, $\beta_2 = 2$, $\beta_3 = 2$, $\beta_4 = -3\sqrt{2}$, and $\beta_j = 0$ for $j > 4$ so that X_4 is independent of Y, even though it is the most significant variable in the joint model in terms of the regression coefficient.

Results. For this example, Table 3.3 shows that MSRc is competitive with other methods in terms of mean true positive rate. However, it has a slightly higher mean false positive rate than that of greedy-INIS and has significantly a higher prediction error than all the other methods. This result is questionable, and hence the data was investigated. The investigation suggested that a different data may have been used by Fan et al. (2009) when they implemented their analysis. In simulating the data, the correlation of X_4 with all other covariates is set to $1/\sqrt{2}$. However, after investigation of the simulated data, nearly all of the sample correlation values is lower than $1/\sqrt{2}$. Figure 3.4 shows the results of this investigation.

Table 3.3: *Mean True Positive Rate and Mean False Positive Rate for Example 4. There are 100 simulations each of size 400, from which the mean true positive rate (TP) and mean false positive rate (FP) were computed. The prediction errors were computed from an independent test data of size 200 for each simulation. The mean prediction error (PE) was computed from the results of 100 simulations. Robust standard deviations are given in parentheses.*

Model	Method	TP	FP	PE
Example 1	MSRc	4.00(0.00)	2.24(0.75)	5.72(0.97)
	INIS	3.99(0.00)	21.96(0.00)	1.62(0.18)
	g-INIS	4.00(0.00)	1.04(1.49)	1.16(0.12)
	penGAM	3.00(0.00)	195.03(21.08)	1.93(0.28)
	ISIS	4.00(0.00)	29.00(0.00)	1.40(0.17)

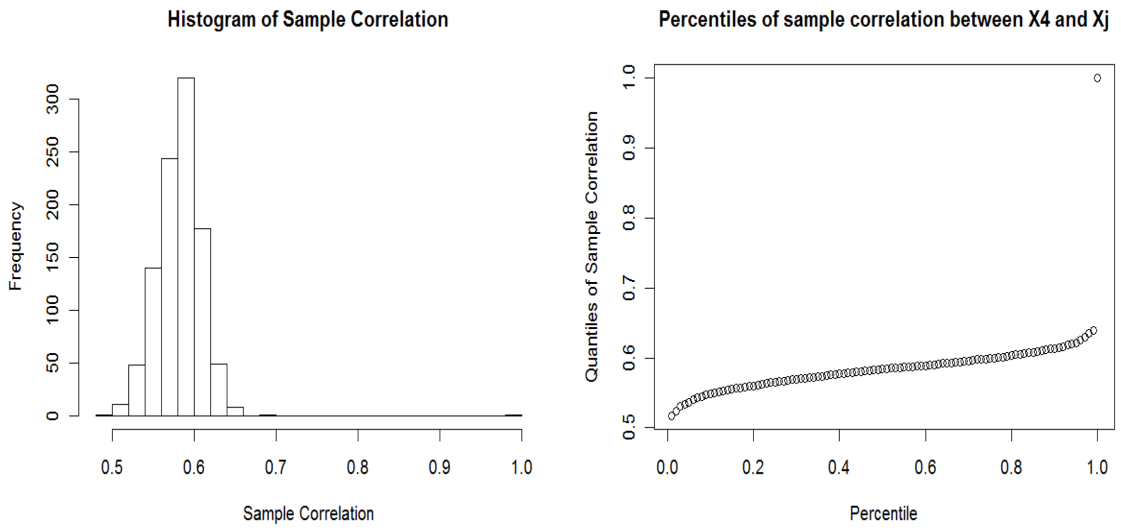
Example 5: Different SNR settings. This simulation is given by Fan et al. (2011) in their paper. The data were generated from the following additive model:

$$Y = 3g_1(X_1) + 3g_2(X_2) + 2g_3(X_3) + 2g_4(X_4) + C\sqrt{3.3843}\varepsilon$$

where the variables $X_1, \ldots, X_p$ are simulated according to Example 2. In this example, C takes a series of different values ($C^2 = 2, 1, 0.5, 0.25$) which lead to the corresponding SNR=0.5,1,2,4. In this example, we set $n = 400$ and $p = 1000$.

Results. The results for INIS and penGAM under different numbers of basis functions, $d_n = 2, 4, 6, 8$, are obtained from the paper of Fan et al. (2011). They did not report the

Figure 3.4: *Sample Correlation between X_4 and Xj. Almost all correlations are smaller than* $1/\sqrt{2} \approx 0.707$



performance of g-INIS or ISIS. From Table 3.4, in which all of the covariates are independent, the MSRc has very good true positive rates under various SNRs. In comparison to INIS and penGAM procedures, the MSRc algorithm has consistently the lowest false selection rates and lowest prediction errors under various SNRs. Table 3.5 presents the more difficult case where pairwise correlation between the covariates is equal to 0.5. MSRc has a competitive performance in terms of true positive rate and has the lowest false selection rate and lowest prediction errors under various SNR values.

Table 3.4: *Mean True Positive Rate and Mean False Positive Rate under Different SNR in Example 5 with t=0. There are 100 simulations, each of size 400, from which the mean true positive rate (TP) and mean false positive rate (FP) were computed. The prediction errors were computed from an independent test data of size 200 for each simulation. The mean prediction error (PE) was computed from the results of 100 simulations. Robust standard deviations are given in parentheses.*

SNR	dn	Method	TP	FP	PE
0.5	NA	MSRc	3.96(0.00)	0.47(0.75)	7.34(0.62)
	2	INIS	3.96(0.00)	2.28(1.49)	7.74(0.79)
		penGAM	4.00(0.00)	27.85(16.98)	8.07(0.92)
	4	INIS	3.93(0.00)	2.29(1.68)	7.90(0.81)
		penGAM	3.99(0.00)	25.61(13.62)	8.21(0.84)
	8	INIS	3.81(0.00)	2.59(2.24)	8.16(1.08)
		penGAM	3.95(0.00)	34.59(20.34)	8.49(0.82)
	16	INIS	3.38(0.75)	2.02(1.49)	8.60(1.13)
		penGAM	3.74(0.00)	33.48(23.88)	9.04(0.93)
1.0	NA	MSRc	4.00(0.00)	0.45(0.75)	3.77(0.50)
	2	INIS	4.00(0.00)	2.16(2.24)	3.98(0.34)
		penGAM	4.00(0.00)	26.51(14.18)	4.20(0.46)
	4	INIS	4.00(0.00)	2.08(1.49)	3.97(0.45)
		penGAM	4.00(0.00)	28.33(15.49)	4.24(0.47)
	8	INIS	4.00(0.00)	2.72(2.24)	4.04(0.43)
		penGAM	4.00(0.00)	36.50(21.83)	4.37(0.47)
	16	INIS	4.00(0.00)	1.80(1.49)	4.26(0.45)
		penGAM	4.00(0.00)	38.60(19.78)	4.80(0.57)
2.0	NA	MSRc	4.00(0.00)	0.51 (0.75)	1.93(0.26)
	2	INIS	4.00(0.00)	2.03(2.24)	2.12(0.17)
		penGAM	4.00(0.00)	25.89(13.06)	2.25(0.24)
	4	INIS	4.00(0.00)	2.38(2.24)	2.06(0.22)
		penGAM	4.00(0.00)	30.37(17.16)	2.21(0.26)
	8	INIS	4.00(0.00)	2.79(2.24)	2.03(0.21)
		penGAM	4.00(0.00)	38.51(16.42)	2.24(0.26)
	16	INIS	4.00(0.00)	1.77(1.49)	2.17(0.25)
		penGAM	4.00(0.00)	42.58(16.60)	2.54(0.30)
4.0	NA	MSRc	4.00(0.00)	0.54 (0.75)	1.01(0.13)
	2	INIS	4.00(0.00)	2.06(2.24)	1.19(0.13)
		penGAM	4.00(0.00)	28.57(14.37)	1.27(0.15)
	4	INIS	4.00(0.00)	2.33(1.49)	1.09(0.10)
		penGAM	4.00(0.00)	30.75(17.35)	1.18(0.14)
	8	INIS	4.00(0.00)	2.88(2.24)	1.02(0.12)
		penGAM	4.00(0.00)	40.51(17.54)	1.14(0.14)
	16	INIS	4.00(0.00)	1.72(1.49)	1.10(0.12)
		penGAM	4.00(0.00)	45.77(19.03)	1.33(0.16)

Table 3.5: *Mean True Positive Rate and Mean False Positive Rate under Different SNR in Example 5 with t=1. There are 100 simulations each of size 400, from which the mean true positive rate (TP) and mean false positive rate (FP) were computed. The prediction errors were computed from an independent test data of size 200 for each simulation. The mean prediction error (PE) was computed from the results of 100 simulations. Robust standard deviations are given in parentheses.*

SNR	dn	Method	TP	FP	PE
0.5	NA	MSRc	2.47(0.75)	0.39(0.75)	7.78(0.73)
	2	INIS	3.35(0.75)	33.67(8.96)	9.49(1.28)
		penGAM	3.10(0.00)	17.74(15.11)	7.92(0.89)
	4	INIS	3.02(0.00)	20.22(2.43)	8.70(1.14)
		penGAM	2.78(0.00)	15.91(10.07)	7.99(0.91)
	8	INIS	2.51(0.75)	10.48(0.75)	8.37(0.89)
		penGAM	2.59(0.75)	16.47(9.70)	8.13(0.90)
	16	INIS	2.10(0.00)	4.47(0.75)	8.44(1.00)
		penGAM	2.41(0.75)	15.56(10.63)	8.42(0.97)
1.0	NA	MSRc	3.61(0.75)	0.19(0.00)	3.87(0.42)
	2	INIS	3.83(0.00)	32.46(9.70)	4.86(0.60)
		penGAM	3.64(0.75)	24.61(21.08)	4.19(0.49)
	4	INIS	3.56(0.75)	20.53(1.68)	4.42(0.52)
		penGAM	3.46(0.75)	22.07(16.04)	4.18(0.49)
	8	INIS	3.09(0.00)	10.67(0.75)	4.28(0.49)
		penGAM	3.12(0.00)	19.92(10.63)	4.30(0.50)
	16	INIS	2.68(0.75)	4.18(0.75)	4.45(0.52)
		penGAM	2.95(0.00)	16.39(11.19)	4.57(0.55)
2.0	NA	MSRc	4.00(0.00)	0.54(0.75)	1.97(0.19)
	2	INIS	3.99(0.00)	29.45(11.57)	2.55(0.38)
		penGAM	3.97(0.00)	36.57(22.57)	2.25(0.28)
	4	INIS	3.93(0.00)	19.12(3.73)	2.26(0.24)
		penGAM	3.91(0.00)	31.31(20.52)	2.19(0.23)
	8	INIS	3.50(0.75)	10.29(0.75)	2.21(0.23)
		penGAM	3.71(0.75)	27.06(19.03)	2.28(0.29)
	16	INIS	2.93(0.00)	4.07(0.00)	2.42(0.32)
		penGAM	3.22(0.00)	19.51(12.13)	2.53(0.30)
4.0	NA	MSRc	4.00(0.00)	0.75(0.75)	1.07(0.11)
	2	INIS	4.00(0.00)	29.47(11.38)	1.45(0.21)
		penGAM	4.00(0.00)	37.27(20.71)	1.27(0.17)
	4	INIS	3.99(0.00)	17.36(5.22)	1.17(0.12)
		penGAM	4.00(0.00)	38.71(20.34)	1.16(0.11)
	8	INIS	3.78(0.00)	10.00(0.00)	1.13(0.16)
		penGAM	3.99(0.00)	41.42(15.86)	1.19(0.13)
	16	INIS	3.02(0.00)	3.98(0.00)	1.36(0.15)
		penGAM	3.72(0.75)	29.58(19.40)	1.43(0.18)

Example 6. This example uses a simulation model discussed in Hall and Miller (2009). The independent variable X_1 has a nonlinear relationship with the response Y and is error contaminated, that is, $X_{i1} = W_i + \delta_i$ and $Y_i = W_i^2 - 1 + \epsilon_i$. Here, W_i is uniformly distributed on [-2,2], and the two error terms δ_i and ϵ_i are both normally distributed with mean equal to 0 and standard deviation equal to 3/4. The other independent variables X_{ij}'s , $j = 2, \ldots, 5000$ were taken to be independent N(0,1). The sample size for this simulation is $n = 200$.

Results. Hall and Miller (2009) implemented this simulation using their generalized correlation procedure (gcorr). Their procedure implemented 500 bootstrap simulations of size $n = 200$, from which they obtained prediction bands for the ranking at $\alpha = 0.02$. Using a cutoff at $\frac{1}{2}p$, Hall and Miller (2009) identified X_{i1} as the variable that has the highest correlation with Y, and there are three false positives. On the other hand, MSRc algorithm showed one true positive and zero false positive.

Example 7. To compare the performance of MSRc and generalized correlation (gcorr) by Hall and Miller (2009) in the presence of multicollinearity, the data generation in this example follows Example 2, with $t = 1$. The pairwise correlation among the independent variables is 0.5. There are 1000 independent variables, and 4 of these are the active variables. The generalized correlation procedure implemented 500 bootstrap simulations of size $n = 400$, from which prediction bands for the ranking at $\alpha = 0.02$ were obtained.

Results. In this example, gcorr used a cutoff at $\frac{1}{2}p$. Table 3.6 shows the mean true positive rate (TP) and mean false positive rate (FP) from 100 simulations. From the results, we conclude that when the independent variables are correlated, gcorr has very poor performance in terms of mean true positive rate and mean false positive rate. Thus, for this simulation example, MSRc is better than gcorr.

Example 8. In this simulation example, the independent variables have pairwise correlation equal to 0.5. To compare the performance of MSRc and generalized correlation (gcorr) by Hall and Miller (2009), the data generation in this example follows Example 3 with $t = 1$.

Table 3.6: *Comparison of MSRc and gcorr: Mean True Positive Rate and Mean False Positive Rate. There are 100 simulations each of size 400, from which the mean true positive rate (TP) and mean false positive rate (FP) were computed. Robust standard deviations are given in parentheses.*

Model	Method	TP	FP
Example 7 t=1	MSRc	3.99(0.00)	0.16(0.00)
	gcorr	2.15(0.00)	10.59(3.73)

There are 1000 independent variables, and 12 of these are the active variables. Generalized correlation procedure implemented 500 bootstrap simulations of size $n = 400$, from which prediction bands for the ranking at $\alpha = 0.02$ were obtained.

Results. Similar to Example 7, the gcorr implemented 500 bootstrap simulations with size $n = 400$, and used a prediction level of $\alpha = 0.02$ and a cutoff at $\frac{1}{2}p$. Table 3.7 shows the mean true positive rate (TP) and mean false positive rate (FP) from 100 simulations. From this comparison, we can see that neither method has the sure screening property. The generalized correlation procedure of Hall and Miller (2009) can only identify one or two active variables out of the 12. On average, the MSRc can identify around 8 active variables. The false positive rate of gcorr is obviously a lot higher than the MSRc. So we conclude that when the independent variables are correlated, gcorr performs poorly in terms of mean true positive rate and mean false positive rate. Hence, MSRc performs better than gcorr.

3.6.2 Real Data Analysis

Enormous amount of data is obtained from genome-wide association studies (GWAS). The goal in GWAS is to understand how genetic variation in an organism, plant or animal is associated with a phenotypic trait such as disease risk and quantitative traits. Typically, its focus is on the detection of single-nucleotide polymorphisms (SNPs) that are associated with a particular phenotypic trait, and on prediction of the latter. Data from GWAS have small n

Table 3.7: *Comparison of MSRc and gcorr: Mean True Positive Rate and Mean False Positive Rate . There are 100 simulations each of size 400, from which the mean true positive rate (TP) and mean false positive rate (FP) were computed. Robust standard deviations are given in parentheses.*

Model	METHOD	TP	FP
Example 8 t=1	MSRc	8.25(1.49)	0.70(0.75)
	gcorr	1.37(0.75)	15.18(2.99)

and large p, have the sparsity nature and the SNP's may be related. In this paper, analysis is done on data of grain yield in wheat from several international trials conducted at the International Maize and Wheat Improvement Center, Mexico. Edited data was downloaded from R package BLR (http://cran.r-project.org/web/packages/BLR/index.html). The phenotype is the average grain yield for each wheat line. The data consist of 599 wheat lines, each genotyped with 1279 DArT markers (Diversity Array Technology). Long et al. (2011) also analyzed this data with the goal of predicting grain yield. They compared the performance of support regression (SVR) models, ε-SVR and least squares SVR, and Bayesian LASSO. In this section, MSRc algorithm is compared with these methods in terms of mean squared error of prediction. For each of ε-SVR and LS-SVR, two types of kernels were considered: a linear kernel and a Gaussian RBF kernel. More information about the implementation of SVR models and Bayesian LASSO is discussed in Long et al. (2011).

The wheat data was partitioned randomly into a training set (480 lines) and a test set (119 lines). Specifically, 480 lines were randomly selected from wheat data to obtain the training set. The remaining 119 lines were considered as the test set. This partitioning was repeated 50 times. Table 3.8 presents the predictive mean squared errors (PMSE) of MSRc, support vector regression (SVR) models and Bayesian LASSO. The values under SVR models and Bayesian LASSO were obtained from the results of Long et al. (2011). The table indicates that MSRc algorithm has superior results in terms of predictive mean squared errors (PMSE). Specifically, the PMSE of MSRc has reduced significantly the PMSE of other methods by at least 43.6% as shown in Table 3.8.

Table 3.8: *Predictive Mean Squared Errors (PMSE) on the Testing Set. Wheat data was partitioned randomly into a training set (480 lines) and a test set (119 lines). This partitioning was repeated 50 times. PMSE for SVR models and Bayesian LASSO are from Long et al. (2011)*

PMSE	MSRc	ε-SVR Linear	ε-SVR RBF	LS-SVR Linear	LS-SVR RBF	Bayesian LASSO
Mean	0.387	0.799	0.686	0.765	0.688	0.768
Standard Errors	0.048	0.086	0.071	0.083	0.079	0.078

Chapter 4

Binary Case: Variable Selection and Prediction

4.1 Introduction

Classification with a binary response variable exist in many fields such as genome wide association studies (GWAS), bioinformatics, microarray, financial data and many more. For example, classification of cancer tissue samples in microarray data has gained popularity in recent years. The greatest challenge of classifying cancer tissue samples is the effective classification given that the sample size (n) is much smaller than the number of covariates or features.

The classical techniques for classification tend to break down in ultrahigh dimensional settings. One example of a classical technique is the Fisher's linear discriminant analysis (LDA). In the study of Bickel and Levina (2004) they have shown that LDA results in poor classification and is asymptotically no better than random guessing. Studies in the literature have stressed the importance of reducing the number of variables before conducting classification to reduce the noise that results in poor classification accuracy. When only a small part of a large set of variables account for the variation of the response variable, using all the variables will only increase misclassification rate. Thus, variable selection plays a very important role in high dimensional classification problems.

A vast number of techniques for variable selection have been developed in recent years. One example is the Features Annealed Independence Rules (FAIR) by Fan and Fan (2008) in which the important features are selected through hard thresholding on two-sample t-statistics and the choice of the optimal thresholding parameter is based on the classification error. Although simulations have shown that FAIR is able to select all important covariates, it has the tendency to result in high false positive rate due to its assumption that all covariates are independent from each other. Fan and Fan (2008) have compared FAIR with nearest shrunken centroids which is another variable selection technique for ultrahigh dimensional settings. Based on simulation studies, the latter procedure selects fewer variables than FAIR but has bigger misclassifications rates. Another variable selection technique is the individual-gene-ranking method (Li et al. (2004) and the references therein) which performs gene selection through a univariate criterion function to provide a list of top ranked covariates. Another procedure which is most recently developed is the BMSF by Zhang et al. (2012). A good property of this procedure is its ability to consider the interaction among covariates during variable selection. However, the implementation of BMSF coupled with classification techniques such as LDA, naive Bayes, SVM and QDA requires extensive computations such that the authors had to present their results by doing variable selection using all the samples rather than using the training data in cross validation. This could lead to poor generalization ability to new data. Although the results of BMSF presented in Zhang et al. (2012) showed high prediction accuracies, the results may not be reproducible in other datasets. In the paper of Zhang et al. (2012) they have also shown that the number of genes selected by BMSF is usually fewer than GeneSrF. For the variables selected by BMSF and GeneSrF, an important issue to consider is whether these two procedures were able to select the true variables and whether they have minimized the number of false variables selected since the authors did not consider simulation study. In general, the main challenge of variable selection techniques is the selection of all the true variables from a large number of covariates while minimizing the number of false variables selected. The development of an algorithm which has high prediction accuracy while using fewer number of covariates is very important in high dimensional settings.

Another issue to consider is the relationship that may exist between the response variable and covariates. Most of the variable selection procedures available are appropriate when there is a linear relationship between the response and each of the covariates. However, variable selection procedures under the nonlinearity assumption is limited. One example of technique that is usually used in this situation is the nonlinear SVM which usually results in high prediction accuracy. In practice, most of the variable selection procedures are combined with nonlinear SVM to improve their performance. However, the problem with SVM procedures is its inclusion of all covariates without providing information to the user on how each variable contributes to the response. The entire classification process is like a black box.

Motivated by the concerns presented, this chapter presents the Most Significant Regression algorithm for the binary data (MSRb). This procedure can be used for two-class classification, and when the relationship between the covariates and the response variable is linear or nonlinear. The main focus of this algorithm is minimizing the false selection rates in variable selection while maintaining high prediction accuracy. Specifically, the procedure is developed for the following two objectives:

1. To select the active independent variables that are significantly associated with the binary response variable;

2. To predict the binary response variable.

The MSRb algorithm is compared to GLMNET through Monte Carlo simulations. In addition, two microarray datasets namely, lung and prostate cancer datasets are also used to compared the performance of MSRb and GLMNET. These two microarray datasets were also used by Zhang et al. (2012) with the application of BMSF and GeneSrF procedures. The results from their paper are also included in this chapter. The significance of MSRb algorithm includes: (1) MSRb often selects a relatively small number of covariates that can accurately classify new observations; (2) MSRb results in better performance than GLMNET in terms of reducing the false positive rate; and (3) It can be used when the relationship between the response variable and the covariates is linear or nonlinear.

Basically, the MSRb algorithm goes through a series of test to select the important variables in a fashion closely related to Least Angle Regression but in the binary setting. Since the response variable is binary, the hypothesis test requires a procedure or test that is suitable for this setting. In the next section, we describe this test in detail. The algorithm is presented in Section 4.3 and the performance measures and numerical measures are presented in Sections 4.4 and 4.5. The application to two cancer microarray datasets are given in Section 4.5.

4.2 NPtest

In this section, we present a nonparametric test (NPtest) to be used in the algorithm. The discussion in this section is restricted to data from one response variable and one covariate. NPtest is a procedure developed by Gharaibeh et al. (2013) which can be used to detect nonlinear relationship between variables. It is a test of heteroscedastic constant regression between a response variable, either discrete or continuous, and a continuous covariate. Specifically, this procedure is testing the hypothesis H_0: $E(y|x)$ does not depend on x against H_1: $E(y|x)$ depends on x. Let (X_j, Y_j), $j = 1, \ldots, N$, be a random sample of the random variables (X, Y). Let the marginal probability density function and cumulative distribution function of X_j be denoted by $f(x)$ and $F(x)$, respectively. In the development of the test statistics, assume that $F(x)$ is differentiable and the fourth conditional central moments of Y_j given $X_j = x$ are uniformly bounded for every x and assume that $\sigma^2(X_i) = Var(Y_i|X_i)$. To test the hypothesis, the test statistic is given as

$$\sqrt{N}(B_N - W_N) \tag{4.2.1}$$

where

$$B_N = \frac{k}{N-1} \sum_{j_1=1}^{N} \left[\frac{1}{k} \sum_{j=1}^{N} Y_j \, g_{Nk}(X_{j1}, X_j) - \frac{1}{Nk} \sum_{j_2=1}^{N} \sum_{j=1}^{N} Y_j \, g_{Nk}(X_{j2}, X_j) \right]^2 + O_p(N^{-1})$$

$$W_N = \frac{1}{N(k-1)} \sum_{j_1=1}^{N} \sum_{j=1}^{N} \left[Y_j \, g_{Nk}(X_{j1}, X_j) - \frac{1}{k} \sum_{j_2=1}^{N} Y_{j2} \, g_{Nk}(X_{j1}, X_{j2}) \right]^2 + O_p(N^{-1}).$$

and $g_{Nk}(X_1, X_2) = I\left(N|\widehat{F}(X_1) - \widehat{F}(X_2)| \le \frac{k-1}{2}\right)$ is the indicator function that the difference between the ranks of X_1 and X_2 is no more than $(k-1)/2$.

Under H_0, the test statistic as $N \to \infty$ is asymptotically normally distributed with mean equal to 0 and variance $\lim_{N\to\infty} \lambda_N$, where

$$\lambda_N = \sum_{j<j'}^{N} E\left\{ \frac{4\sigma^2(X_j)\sigma^2(X_{j'})}{N(k-1)^2} \left[[k - |j'_* - j_*|]^2 + [k - |j'_* - j_*|] - 2I\left(|j'_* - j_*| \le \frac{k-1}{2}\right) \right] + O(N^{-1}) I(|j'_* - j_*| \le k-1) \right. \quad (4.2.2)$$

and j'_*, j_* are the ranks of $X_{j'}$ and X_j among the covariate values $\mathbf{X} = (X_1, \ldots, X_N)$.

For detailed discussion of NPtest, refer to Gharaibeh et al. (2013). For the MSRb algorithm, the NPtest is used to test the nonlinear relationship between the current working residual and each of the covariates. The p-values corresponding to the test statistics are then used to determine which covariate will be selected. The variable is selected based on its p-value and the criteria which is explained in the next section.

4.3 Most-Significant-Regression Algorithm, MSRb

Suppose that we have a random sample $(\mathbf{X}_i, Y_i)$, $i = 1, 2, ..., n$ observed from an unknown population $(\mathbf{X}, Y)$, where $\mathbf{X} = (X_{1i}, \ldots, X_{pi})^T$. We consider the case that $p >> n$. Suppose that only a small subset of covariates of size p' contribute to the response and $p' < p$. For convenience of notation, we denote these p' covariates as $\mathbf{Z} = (X_{j_1}, \ldots, X_{j_{p'}})^T$. Moreover, let the true model be

$$\begin{aligned} logit(p(Y=1|\mathbf{Z})) &= log\frac{p(Y=1|\mathbf{Z})}{1-p(Y=1|\mathbf{Z})} \\ &= m(\mathbf{Z}), \end{aligned} \tag{4.3.1}$$

The covariates in $\mathbf{Z}$ are called active variables which we want to identify from the entire set of covariates $\Omega = \{X_1, \ldots, X_p\}$. Let $\mathbf{X}^*$ be the set of variables selected to enter the model. Initialize $\mathbf{X}^* = \phi$.

4.3.1 Variable Selection Algorithm

For variable selection, we use the training data. Let n_{tr} be the number of observations in the training data, a be the threshold of the p-value for filtering out the most irrelevant variables, k be the number of active variables already selected at the current stage and r_i be the current working residual for the i^{th} observation with the active variables in the model. Denote $\widehat{g}_i$ as the current fitted value in the logit scale for the i^{th} observation.

Step 0: Initialize $a = 0.5$, $k = 0$, $r_i = Y_i$ and $\widehat{g}_i = 0$.

Step 1: Group the values of each independent variable X_j according to the value of the response variable Y, $j = 1, ..., p$. The response variable has two classes and therefore there are 2 groups for the values of X_j. Conduct a two sample t-test for the 2 groups of X_j values and obtain the p-value. Since there are p independent variables at the start of the algorithm, then there are p two sample t-tests giving p p-values.

Step 2: Discard all independent variables with p-value$>a$ and keep those that have p-values$<a$ and put them into set $\mathbf{X_{new}}$. Suppose there are m variables in $\mathbf{X_{new}}$. If $\mathbf{X_{new}} = \phi$, terminate the algorithm. Otherwise, proceed to the next step.

Step 3: Conduct a Bonferroni correction procedure at significance level equal to 0.01 for all variables in $\mathbf{X_{new}}$. That is, select all variables in $\mathbf{X_{new}}$ with p-values less than $0.01/m$ and put them in set $\mathbf{X}^*$. Update k, which is equal to the number of variables in $\mathbf{X}^*$. If $k = 0$, proceed to Step 5. Otherwise, update $\mathbf{X_{new}} \Leftarrow \Omega - \mathbf{X}^*$ and proceed to Step 4.

Step 4: Let the set of all selected covariates be $\mathbf{X}^* = \{X_{i_1}, X_{i_2}, ..., X_{i_k}\}$. Enter each variable into the model one at a time as follows. Let j be the index of how many covariates are already in the model.

(1) Initialize $j = 1$. Using the first covariate, X_{i_1}, fit a nonlinear logistic model $logit(p(Y_i = 1|X_{i_1} = x)) = f_1(x)$, where $f_1(x)$, is estimated by a smooth function $\widehat{f_1}$ obtained via thin-plate smoothing spline. From the fitted model, obtain the fitted value for each observation in the logit scale and denote it as $\widehat{\eta_i} = \widehat{f_1}(X_{i_1 i})$, $i = 1, ..., n_{tr}$. The estimate for the smooth function is obtained using the generalized additive model in the MGCV package by Wood (2008). Thin-plate smoothing spline is implemented since it is the optimal smoother of any given basis dimension or rank according to Wood (2003). For MSRb algorithm, the default basis dimension was used which is equal to 8 when the model has 1 single variable.

(2) For all $i = 1, ..., n_{tr}$, do the following: calculate the weights $w_i = \frac{e^{\widehat{\eta_i}}}{(1+e^{\widehat{\eta_i}})^2}$; then compute the working residual $r_i = \frac{y_i - \widehat{\eta_i}}{w_i}$ and obtain the adjusted dependent variable $z_i = \widehat{\eta_i} + r_i$; update $\widehat{g_i} \Leftarrow \widehat{g_i} + \widehat{\eta_i}$.

(3) Update $j \Leftarrow j + 1$ and fit a weighted nonlinear regression model between the j^{th} covariate $X_{i_j} \epsilon \mathbf{X}^*$ and the current adjusted dependent variable Z with observations $z_1, z_2, ..., z_{n_{tr}}$. Obtain the following for all $i = 1, ..., n_{tr}$: the new fitted values $\widehat{\eta_i}$, the new weights $w_i = \frac{e^{\widehat{\eta_i}}}{(1+e^{\widehat{\eta_i}})^2}$, the new working residual $r_i = \frac{z_i - \widehat{\eta_i}}{w_i}$, and the new adjusted dependent variable $z_i = \widehat{\eta_i} + r_i$. Update $\widehat{g_i} \Leftarrow \widehat{g_i} + \widehat{\eta_i}$.

(4) Repeat (3) for all j such that $j \leq k$.

Step 5: From the set $\mathbf{X_{new}}$, select the variable with smallest p-value.

Step 6: Update $k \Leftarrow k + 1$. Denote the previously selected variable as X_{i_k}. Update $\mathbf{X_{new}} \Leftarrow \mathbf{X_{new}} - X_{i_k}$ and perform the following:

If $k = 1$, do the following: Using the covariate X_{i_1}, and the current working residual $r = Y$ as the response variable to fit a nonlinear logistic model $logit(p(Y = 1|X_{i_k} = x)) =$

$f_k(x)$, where $f_k(x)$ is a smooth function to be estimated by $\widehat{f}_k(x)$ obtained via thin-plate smoothing spline using the generalized additive model in MGCV package by Wood (2008). From the fitted model, obtain the following updates for $i = 1, ..., n_{tr}$: the fitted values $\widehat{\eta}_i = \widehat{f}_k(X_{i_k i})$; the weights equal to $w_i = \frac{e^{\widehat{\eta}_i}}{(1+e^{\widehat{\eta}_i})^2}$; the working residual $r_i = \frac{y_i - \widehat{\eta}_i}{w_i}$; and the adjusted dependent variable $z_i = \widehat{\eta}_i + r_i$.

If $k > 1$, do the following: Fit a weighted nonlinear regression model between the covariate X_{i_k} and the current working residual $(r_1, ..., r_{n_{tr}})$ as the response variable. For $i = 1, ..., n_{tr}$, obtain the fitted values from the fitted model and denote it as $\widehat{\eta}_i$; compute the weights $w_i = \frac{e^{\widehat{\eta}_i}}{(1+e^{\widehat{\eta}_i})^2}$; update the working residual $r_i \Leftarrow \frac{r_i - \widehat{\eta}_i}{w_i}$; and compute the adjusted dependent variable $z_i = \widehat{\eta}_i + r_i$.

Step 7: Update $\widehat{g}_i \Leftarrow \widehat{g}_i + \widehat{\eta}_i$ and do the following: Fit a logistic regression model between the response variable Y and a surrogate variable g using observation pairs $(Y_1, \widehat{g}_1), ..., (Y_{n_{tr}}, \widehat{g}_{n_{tr}})$:

$$logit(p(Y = 1|\widehat{g})) = \beta_0 + \beta_1 g,$$

where the model estimate is obtained using generalized additive model in the MGCV package by Wood (2008). Then from the fitted model, obtain the p-value p_c from the deviance test of β_1 equal to zero.

Step 8: Conduct an NPtest between the current working residual $r_1, ..., r_{n_{tr}}$ and each of the variables in $\mathbf{X_{new}}$. From the NPtest, obtain all p-values p_{bi}, $i = 1, ..., m$. If there is no p-value close enough to p_c, that is, $|p_{bi} - p_c| \geq 0.00005$, for all i, terminate the algorithm and report the variables in $\mathbf{X}^*$ as the variables selected. Otherwise, select the independent variable whose p-value is close enough to p_c and then return to Step 6.

4.3.2 Model Building and Prediction Algorithm

From the previous section, the variables are selected using a training data. Denote the set of variables selected as $\mathbf{X}^*$. From the same training data, model building is implemented. The model derived from the training data is then used for making predictions of the test data. Let n_{te} be the number of observations in the test data. Specifically, the following steps are done to obtain the final model and prediction of the test data.

Step 1: Let $\widehat{h}_i$ be the storage for the predicted value of the i^{th} observation of the test data in the logit scale . The covariates are entered to the model one at a time. Let j be the index of covariates entered in the model. Initialize $\widehat{h}_i = 0$ and j=1.

Step 2: Using the training data, fit a nonlinear logistic model of Y and covariate X_{i_j} :

$$logit(p(Y_i = 1|X_{i_j} = x)) = f_j(x),$$

where $f_j(x)$ is estimated by a smooth function $\widehat{f}_j(x)$ obtained via thin-plate smoothing spline using generalized additive model in MGCV package by Wood (2008). From the fitted model, obtain the fitted values $\widehat{\eta}_i = \widehat{f}_j(X_{i_j i})$. For all $i = 1, ..., n_{tr}$, compute the weights equal to $w_i = \frac{e^{\widehat{\eta}_i}}{(1+e^{\widehat{\eta}_i})^2}$, the working residual $r_i = \frac{y_i - \widehat{\eta}_i}{w_i}$ and the adjusted dependent variable $z_i = \widehat{\eta}_i + r_i$.

Step 3: Using the model built in Step 2, predict the response variable in the logit scale for each observation in the test data using the same covariate used in modeling and denote the predicted value from this model as $\widehat{\upsilon}_i$, $i = 1, ..., n_{te}$. Update $\widehat{h}_i \Leftarrow \widehat{h}_i + \widehat{\upsilon}_i$.

Step 4: Update $j \Leftarrow j + 1$. Using the training data, fit a weighted nonlinear regression model between the current working residual r_i and j^{th} covariate in $\mathbf{X}^*$. Obtain the following for $i = 1, ..., n_{tr}$: the new fitted values $\widehat{\eta}_i$, the new weights $w_i = \frac{e^{\widehat{\eta}_i}}{(1+e^{\widehat{\eta}_i})^2}$, the new working residual $r_i = \frac{z_i - \widehat{\eta}_i}{w_i}$ and the new adjusted dependent variable $z_i = \widehat{\eta}_i + r_i$.

Step 5: Using the model built in Step 4, predict the response variable for the test data given X_{i_j} and denote as $\widehat{\upsilon}_i$, $i = 1, ..., n_{te}$. Update $\widehat{h}_i \Leftarrow \widehat{h}_i + \widehat{\upsilon}_i$.

Step 6: Repeat Steps 4 and 5 until all the covariates in $\mathbf{X}^*$ are used in model building. The $\widehat{h}_i$ is the accumulation of all predicted values in the logit scale as the covariates in $\mathbf{X}^*$ are entered into the model one at a time in predicting the successive working residual for the test data.

Step 7: The predicted value for the i^{th} observation in the test dataset in the response scale denoted by $\widehat{y}_i$ is now obtained by computing

$$\widehat{y}_i = \begin{cases} 1, & \text{if } \frac{e^{\widehat{h}_i}}{1+e^{\widehat{h}_i}} \geq 0.5 \\ 0, & \text{otherwise .} \end{cases} \tag{4.3.2}$$

4.4 Performance Measures

Cross validation (CV) was implemented to evaluate the performance of the algorithm. The data was partitioned randomly into a training set and a test set. For k-fold CV, the data was randomly partitioned into k folds in which the test set consists of the observations from the m^{th} fold and the training set consists of the remaining observations, $m = 1, ..., k$. Variable selection and model building are implemented using the training data while prediction is obtained for the test data. The MSRb algorithm is applied k times in k-fold CV since there k different test data and k different training data. In the end of the algorithm, predicted values of test data from all k folds are combined to give the prediction accuracy.

The results from different runs of CV maybe different due to the random partition. Hence, we repeat with r runs. For examples with simulated data, both data generation and k-fold cross validation were done r times. For real datasets, k-fold cross validation was done r times.

Before we present the performance measures used to compare MSRb with existing procedures, we first define some terms: (1) active variables are the independent variables used to generate the true model, while inactive variables are the independent variables not used in the true model; (2) accuracy is the percentage of correctly classified samples in the combined predicted values of test data in CV.

To compare the performance of MSRb with existing procedures, we use the mean true positive rate (TP), mean false positive rate (FP) and mean accuracy. Specifically, the following measures are computed:

$$TP = \frac{\sum_{i=1}^{r} \text{number of active variables recruited in the } i^{th} \text{ run}}{r} \quad ,$$

$$FP = \frac{\sum_{i=1}^{r} \text{number of inactive variables recruited in the } i^{th} \text{ run}}{r},$$

$$MeanAccuracy = \frac{\sum_{i=1}^{r} \text{percentage of correct classifications in the } i^{th} \text{ run}}{r},$$

where r is the number of runs. Again, note that variable selection and model building are based on training data while prediction is done using the test data.

4.5 Numerical Comparisons

4.5.1 Simulation Models and Results

To demonstrate the power of our proposed MSRb method and to be able to compare with GLMNET, we present comparison results for 3 data generation settings.

Example 1. Covariates are generated from the normal distribution with mean 0 and standard deviation equal to 1. That is, $X_i \sim$ iid N(0,1), $i = 1, 2, ...p$. The response Y is simulated from the Bernoulli distribution with log odds of success given by the function $f(x_1, x_2) = e^{2X_1} + e^{2X_2} - \frac{16}{5}$. Based on the function, there are two active variables, namely, X_1 and X_2. We generate random pairs of $(\mathbf{X}, Y)$ of sample size $n = 60, 100, 200, 300$ and $p = 1000$. Variable selection and response estimation was done using 3-fold cross-validation (CV). Data were simulated 100 times. This example illustrates the performance of MSRb when the active covariates are not linearly related to the log odds of the success probability of the response.

Results. For this example, Table 4.1 shows that the mean accuracy of MSRb is slightly lower than GLMNET by less than 1% when the sample size is 60 and 100. However, when the sample size is increased to 200 and 300, MSRb becomes more accurate by approximately 7% than GLMNET. In terms of true positive rate, GLMNET is slightly higher than MSRb by 0.4 for sample size 60 and by 0.7 for sample size 100. When the sample size is increased to 200 and 300, the two procedures have comparable true positive rates. As regards to false positive rate, MSRb is significantly lower than GLMNET for all sample sizes considered in this simulation. MSRb has almost zero false positive rates while GLMNET's average false positive rate ranges from 7.55 to 26.06.

Table 4.1: *Performance Measures for Example 1. With* $p = 1000$*, sample sizes of 60, 100, 200, and 300 were simulated 100 times, from which the mean true positive rate (TP), mean false positive rate (FP) and mean accuracy were computed. These performance measures were obtained via 3-fold CV. Robust standard deviations are given in parentheses.*

Method	Size (n)	Mean Accuracy	TP	FP
MSRb	60	61.87 (0.09)	0.61(0.75)	0.39 (0.75)
GLMNET		62.82 (0.09)	1.07 (1.49)	7.55 (9.20)
MSRb	100	71.58 (0.07)	1.10 (0.00)	0.08 (0.00)
GLMNET		72.34 (0.07)	1.86 (0.00)	15.62 (12.38)
MSRb	200	85.12 (0.03)	1.96 (0.00)	0.00 (0.00)
GLMNET		77.81 (0.03)	2.00 (0.00)	18.23 (15.86)
MSRb	300	86.32 (0.02)	2.00 (0.00)	0.01 (0.00)
GLMNET		78.81 (0.03)	2.00 (0.00)	26.06 (20.83)

Example 2. Covariates are generated from the normal distribution with mean 0 and standard deviation equal to 1. That is, $X_i \sim$ iid N(0,1), $i = 1, 2, ...p$. The response Y is simulated from the Bernoulli distribution with log odds of success given by the function $f(x_1, x_2) = -4 + \frac{8}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{X_1}{2} + \frac{X_2}{2})^2}$. Based on the function, there are two active variables, namely, X_1 and X_2. We generate random pairs of $(\mathbf{X}, Y)$ of sample size $n = 60, 100, 200, 300$ and $p = 1000$. Variable selection and response estimation was done using 3-fold cross-validation (CV). Data were simulated 100 times. This example illustrates the performance of MSRb when the contribution of the active covariates to the log odds of the success probability of the response is nonlinear and non-additive.

Results. For this example, Table 4.2 shows that the mean accuracy of MSRb is slightly lower than GLMNET by 2% when the sample size is 60. However, as the sample size increases, MSRb becomes more accurate than GLMNET. Specifically, for sample sizes 100 and 300, MSRb is higher in accuracy by approximately 4% and 1% respectively. When the sample size is 200, the prediction accuracy of the two procedures are comparable. In terms of true positive rates, GLMNET is slightly better than MSRb by approximately 0.5 for sample size 60 and 100. When the sample size increases to 200 and 300, the two procedures have comparable true positive rates. As regards to false positive rates, MSRb is significantly lower than GLMNET for all sample sizes considered in the simulation. MSRb has almost zero mean false positive rate for all sample sizes while GLMNET's mean false positive rate falls from 7.17 to 20.91.

Table 4.2: *Performance Measures for Example 2. With $p = 1000$, sample sizes of 60, 100, 200, and 300 were simulated 100 times, from which the mean true positive rate (TP), mean false positive rate (FP) and mean accuracy were computed. These performance measures were obtained via 3-fold CV. Robust standard deviations are given in parentheses.*

Method	Size (n)	Mean Accuracy	TP	FP
MSRb	60	57.98 (0.10)	0.41 (0.75)	0.59 (0.75)
GLMNET		60.25 (0.07)	0.84 (1.49)	7.17 (9.33)
MSRb	100	71.58 (0.07)	1.07 (0.00)	0.90 (0.00)
GLMNET		67.35 (0.08)	1.69 (0.50)	14.00 (13.18)
MSRb	200	74.91 (0.06)	1.68 (0.75)	0.03 (0.00)
GLMNET		74.69 (0.04)	2.00 (0.00)	18.21 (14.05)
MSRb	300	77.83 (0.02)	1.98 (0.00)	0.01 (0.00)
GLMNET		76.34 (0.03)	2.00 (0.00)	20.91 (18.84)

Example 3. Covariates are generated from the normal distribution with mean 0 and standard deviation equal to 1. That is, $X_i \sim$ iid N(0,1), $i = 1, 2, ...p$. The response Y is simulated from the Bernoulli distribution with log odds of success given by the function $f(x_1) = 1.5X_1$. Based on the function, there is only one active variable, namely, X_1. We generate random pairs of $(\mathbf{X}, Y)$ of sample size $n = 60, 100, 200, 300$ and $p = 1000$. Variable selection and response estimation was done using 3-fold cross-validation (CV). Data were simulated 100 times. This example illustrates the performance of MSRb when the active covariate is linearly related to the log odds of the success probability of the response

Results. For this example, Table 4.3 shows that the mean accuracy of MSRb is higher than GLMNET for all sample sizes. Specifically, the prediction accuracy of MSRb is higher than GLMNET by approximately 1% when the sample size is 60 and 300, and approximately 2.5% when the sample size is 100 and 200. In terms of true positive rate, the two procedures have comparable result. As regards to false positive rate, MSRb is consistently lower than GLMNET for all sample sizes considered in the simulation. Specifically, the mean false positive rate for MSRb ranges from 0.04 to 0.40. On the other hand, the mean false positive rate for GLMNET ranges from 5.78 to 14.19.

Example 4. In real datasets such as in bioinformatics, sample size is usually smaller than 400 and the number of covariates is at least 10000. To address this scenario, Examples 1, 2 and 3 were used to simulate data with sample size (n) equal to 60 and 300, while the number of covariates (p) is set to 12000.

Results. The data generation, variable selection and model building with training data, and prediction for the test data in CV are repeated 10 times. The average accuracy, average TP, average FP from the 10 runs with robust standard deviations are reported in Table 4.4.

For Example 1, the mean accuracy of MSRb is higher than GLMNET by approximately 2% and 7% for sample sizes 60 and 300 respectively. In terms of true positive rate, MSRb is lower by 0.40 when the sample size is 60. The two procedures have comparable true positive rates when $n = 300$. For the false positive rate, MSRb is significantly lower than of

Table 4.3: *Performance Measures for Example 3. With $p = 1000$, sample sizes of 60, 100, 200, and 300 were simulated 100 times, from which the mean true positive rate (TP), mean false positive rate (FP) and mean accuracy were computed. These performance measures were obtained via 3-fold CV. Robust standard deviations are given in parentheses.*

Method	Size (n)	Mean Accuracy	TP	FP
MSRb	60	64.43 (0.14)	0.61 (0.75)	0.40 (0.75)
GLMNET		63.08 (0.09)	0.67 (0.75)	5.78 (6.65)
MSRb	100	70.72 (0.07)	0.89 (0.00)	0.16 (0.00)
GLMNET		67.09 (0.08)	0.96 (0.00)	11.33 (12.38)
MSRb	200	73.53 (0.03)	1.00 (0.00)	0.06 (0.00)
GLMNET		71.63 (0.03)	1.00 (0.00)	13.81 (14.61)
MSRb	300	73.44 (0.02)	1.00 (0.00)	0.04 (0.00)
GLMNET		72.41 (0.03)	1.00 (0.00)	14.19 (12.75)

GLMNET. The false positive rate of MSRb is almost zero while GLMNET has mean false positive rates ranging from 17.53 to 22.13.

For Example 2, the mean accuracy of GLMNET is higher by 5% than MSRb when the sample size is 60. When the sample size is increased to 300, GLMNET has higher accuracy by 1%. In terms of true positive rates, GLMNET is higher by 0.6 than MSRb for sample size $n = 60$, and they have comparable result when $n = 300$. In terms of false positive rate, MSRb has almost zero false positive rate while GLMNET has around 21 for both sample sizes $n = 60$ and $n = 300$.

For Example 3, MSRb has higher accuracy than GLMNET for both sample sizes 60 and 300. MSRb is higher in accuracy by 1% and 2% when $n = 60, 300$ respectively. The true positive rates of both procedures are comparable for sample sizes 60 and 300. In terms of false positive rate, MSRb has significantly lower false positive rate than GLMNET. Specifically, the average false positive rate for MSRb is between 0.27 and 0.70 for $n = 60$, while that for GLMNET is at least 17. When the sample size is $n = 300$, the average false positive rate for MSRb is almost zero, while that of GLMNET is between 15.56 and 22.13.

In general, Table 4.4 showed that when $p = 12000$, MSRb has better prediction accuracy than GLMNET for all examples except for Example 2 with $n = 60$. In terms of true positive rate, GLMNET is better in selecting the true variables when the sample size is small (n=60). When the sample size is 300, MSRb and GLMNET both select the correct active variables. However, in terms of false positive rate, MSRb is significantly better for all examples. In addition, the standard deviations of mean false positive rate for GLMNET are very large. It is also important to note that for both methods, MSRb and GLMNET showed improvement in mean accuracy, true positive rate and false positive rate as the sample size increases.

Table 4.4: *Performance Measures with* $p = 12000$, *and* $n = 60, 300$ *from simulation of size 10. The mean true positive rate (TP), mean false positive rate (FP) and mean accuracy were computed. These performance measures were obtained via 3-fold CV. Robust standard deviations are given in parentheses.*

		MSRb			GLMNET		
Example	n	Mean Accuracy	TP	FP	Mean Accuracy	TP	FP
Example 1	60	68.00 (0.00)	0.73 (0.56)	0.27 (0.56)	65.67 (0.00)	1.13 (1.49)	17.53 (16.98)
	300	85.90 (0.02)	2.00 (0.00)	0.00 (0.00)	78.3 (0.02)	2.00 (0.00)	22.13 (16.23)
Example 2	60	58.67 (0.00)	0.30 (0.75)	0.70 (0.75)	63.17 (0.00)	0.97 (1.49)	20.67 (29.29)
	300	77.43 (0.02)	2.00 (0.00)	0.00 (0.00)	76.33 (0.03)	2.00 (0.00)	21.43 (22.01)
Example 3	60	59.33 (0.00)	0.43 (0.75)	0.57 (0.75)	58.83 (0.00)	0.67 (0.75)	17.47 (19.59)
	300	74.17 (0.01)	1.00 (0.00)	(0.06) (0.00)	72.05 (0.02)	1.00 (0.00)	15.56 (14.55)

4.5.2 Real Data Analysis

A common task in bioinformatics is the selection of relevant genes, where researchers try to determine the smallest possible set of genes that can still achieve good predictive performance. A typical data in bioinformatics consist of small sample size (n) and very large number of genes. In this section, we present the results of MSRb to two gene-expression datasets related to lung and prostate tumors. The lung cancer dataset includes 12533 genes with sample size equal to 181, among which 150 belong to class 1 and 31 belong to class 2. On the other hand, the prostate data involves 12626 genes with sample size equal to 33, among which 24 are from class 1, and 9 from class 2.

The two datasets were used to compare MSRb with GLMNET. Ten-fold cross validation as explained in Section 4.4 was done using 10 runs for each of the datasets and the mean accuracies with robust standard deviations were obtained to compare the performance of the different procedures. Table 4.5 indicates that MSRb algorithm for the prostate data has comparable accuracy with GLMNET. On the other hand, for the lung dataset, MSRb has slightly lower accuracy by 2% than GLMNET. In terms of the number of genes selected, MSRb is using only one covariate for both datasets while GLMNET uses 14 genes for prostate data and 22 genes for lung data.

Zhang et al. (2012) also analyzed these two datasets. The results of their analysis for BMSF-SVM, GeneSrF-SVM, BMSF-NB, GeneSrF-NB, BMSF-LDA, GeneSrF-LDA, BMSF-QDA, and GeneSrF-QDA are included in the bottom portion of Table 4.5. The authors have implemented these procedures in a different manner. They first selected relevant genes from the entire microarray data using all samples with BMSF and GeneSrF. After selecting the genes, they conducted 10-fold cross validation in the data where 9 folds are used in SVM, NB, LDA, and QDA to build a model while 1 fold is used as a test data. Prediction of the response for the test data uses the model which was developed from the training data. Modeling and prediction was implemented 10 times because there are 10 folds. When the m^{th} fold was treated as test data, the remaining observations are treated as training data,

$m = 1, ..., 10$. The prediction of test data for each of the 10 folds are combined and the accuracy is computed which is the percentage of correct classifications. This procedure is repeated in 10 runs and the average accuracy is the mean percentage of correct classifications in 10 runs. On the other hand, the results of MSRb and GLMNET are obtained by selecting the relevant genes from the training data, then build a model from the same training data and then predict the class of test data. For this reason, the results from Zhang et al. (2012) cannot be compared directly with the results of MSRb and GLMNET.

In the prostate data, the prediction accuracy of MSRb is the same with GLMNET, BMSF-NB, GeneSrF-NB and BMSF-QDA. On the other hand, MSRb is higher in accuracy by approximately 1 to 2 % than BMSF-SVM, BMSF-LDA, GeneSrF-QDA. The other procedures GeneSrF-SVM and GeneSrF-LDA have accuracies that are much lower than MSRb by 6%. In terms of the number of genes selected, MSRb is using only one gene to achieve an accuracy of 99.69% while other procedures are using 2 to 3 genes. In the lung data, the MSRb has the lowest accuracy equal to 96.30% while the other procedures has prediction accuracy between 97.56% and 99.11%. However, the MSRb procedure is using only 1 gene for prediction while the other procedures are using 8 to 22 genes. In general, MSRb is using the fewest number of genes while providing high prediction accuracy. Furthermore, the accuracy of Zhang et al. (2012) may not be achievable for a new data because of overfitting as a result of variable selection using the entire set of samples. On the other hand, MSRb and GLMNET reported accuracy is closer to the generalization accuracy for new data.

Table 4.5: *Mean accuracy with robust standard deviations of different procedures obtained from 10-fold CV with 10 runs. Performance measures for BMSF and GeneSrF are from Zhang et al. (2012).*

Procedure	Prostate		Lung	
	Number of Genes	Mean Accuracy	Number of Genes	Mean Accuracy
MSRb	1	99.69 (0.00)	1	96.30 (0.01)
GLMNET	14	99.69 (0.00)	22	98.51 (0.00)
BMSF-SVM	2	98.48 (1.59)	8	99.11 (0.64)
GeneSrF-SVM	3	93.93 (2.47)	12	98.95 (0.40)
BMSF-NB	2	99.69 (0.95)	8	98.39 (0.31)
GeneSrF-NB	3	99.69 (0.95)	12	98.34 (0.00)
BMSF-LDA	2	96.66 (0.95)	8	97.79 (0.26)
GeneSrF-LDA	3	93.93 (0.00)	12	98.34 (0.00)
BMSF-QDA	2	100.00 (0.00)	8	97.56 (0.46)
GeneSrF-QDA	3	96.66 (1.72)	12	98.34 (0.26)

Chapter 5

Summary and Post-dissertation Research

5.1 Summary

This dissertation developed algorithms for variable selection and response variable estimation when the response variable is continuous or binary. The development of these algorithms are inspired by the Least Angle Regression. Both MSRb and MSRc are multi-step model selection algorithm to select variables in sparse ultrahigh dimensional additive models. The variables go through a series of nonlinear dependence evaluation. The algorithm can be used when the predictors are linearly or nonlinearly related to the response.

Generalized correlation procedure (gcorr) and NIS based procedures such as INIS and g-INIS are very efficient variable selection procedures for continuous case. NIS-based procedures are coupled with low dimensional techniques such as penGAM and SCAD to improve their results. Although these procedures are very good in terms of prediction error and true positive rate, they tend to have high false positive rate. In the simulations study, we find that MSRc algorithm is competitive with NIS-based procedures in terms of prediction error and true positive rate. Also, simulations showed that MSRc has significantly less false positive rate than generalized correlation procedure (gcorr). The advantage of using

MSRc over NIS-based procedures and gcorr is its capability to reduce false positive rate. NIS-based procedures and gcorr both select variables based on the marginal contribution of the covariates with the response variable. On the other hand, MSRc algorithm selects only the first variable by looking at the marginal contribution of covariates and it selects the next variable based on its contribution on the response variable conditional on the active variables which are already selected in the previous steps. However, MSRc has the tendency to fail when the first variable selected is a false variable.

Another part of this dissertation is the development of a variable selection and response estimation when the response variable is binary. This algorithm is called MSRb which is a modified version of MSRc. From our simulated data, MSRb is competitive with GLMNET in terms of mean accuracy and true positive rate. However, in terms of false positive rate, MSRb is much better than GLMNET. Real studies using two microarray data (lung and prostate data) showed that MSRb is competitive with BMSF and GeneSrF. In terms of the numbers of variables selected. MSRb has fewer number of covariates selected.

MSR algorithm is different from forward stepwise selection in the sense that the latter procedure involves testing the addition of each covariate using a chosen model comparison criterion, adding any one variable that improves the model the most, and repeating this process until none improves the model. The addition of a variable may be based on any of the following criteria: F-tests, adjusted R-square, Akaike information criterion, Bayesian information criterion, and Mallows's Cp. On the other hand, MSR algorithm adds a variable only if its contribution is as much as the contribution of the variables that are already in the model, and this is done by evaluating which of the covariates has p-value closest to p_c which is obtained from relating the response variable Y and the current fitted values.

5.2 Post-dissertation Research

The algorithms developed in this dissertation are limited to situations where the response variable is continuous or binary. The algorithm could be extended to the case where the response variable follows a multinomial distribution in which the number of classes is more than two.

In addition, MSRb and MSRc algorithms could not delete a variable that has already been selected in previous steps. These algorithms could be modified in a way that the active variables may be removed in the next steps.

Additional research can also be considered to quantify the conditions under which MSR algorithm could perform well. The theoretical properties may be pursued in future research.

Bibliography

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, pages 1545–1588.

Austin, P. and Lee, D. (2011). Boosted classification trees result in minor to modest improvement in the accuracy in classifying cardiovascular outcomes compared to conventional classification trees. *American Journal of Cardiovascular Disease*, 1(1):1–15.

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

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*. World Scientific Pub Co Inc.

Candes, E. and Tao, T. (2007). The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35:2313.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2, 3, Article 27.

Cho, H. and Fryzlewicz, P. (2012). High dimensional variable selection via tilting. *Journal of the Royal Statistical Society*, 74:593–622.

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

Diaz-Uriarte, R. and Alvarez de Andres, S. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(3).

Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425455.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *Proceedings of the 7th International Conference on Information and Knowledge Management.*

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, 32:407–499.

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

Fan, J., Feng, Y., and Song, R. (2011). Nonparametric independence screening in sparse ultra-high-dimensional additive models. *Journal of the American Statistical Association*, 106(494):544–557.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96:1348–1360.

Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society*, 70:849911.

Fan, J., Samworth, R., and Wu, Y. (2009). Ultra-dimensional variable selection via independent learning: Beyond the linear model. *Journal of Machine Learning Research*, 10:18291853.

Frank, I. and Friedman, J. (1993). A statistical view of some chemometrics regression tools (with discussion). *Technometrics*, 35:109–148.

Friedman, J., Hastie, T., and Tibshirani, R. (2009). Regularization paths for generalized linear models via coordinate descent.

Gharaibeh, M., Sahtout, M., and Wang, H. (2013). A nonparametric lack-of-fit test of constant regression in presence of heteroscedastic variances. Manuscript in revision.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422.

Hall, P., Marron, J., and Neeman, A. (2005). Geometric representation of high dimension, low sample size data. *Journal of the Royal Statistical Society*, B(67):427–444.

Hall, P. and Miller, H. (2009). Using generalised correlation to effect variable selection in very high dimensional problems. *Journal of Computational and Graphical Statistics*, 18:533–550.

Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. Chapman and Hall, London.

Ho, T. K. (1995). Random decision fores. *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, pages 278–282.

HoerlA.E. and R.W., K. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67.

Huang, J., Horowitz, J., and Wei, F. (2010). Variable selection in nonparametric additive models. *The Annals of Statistics*, 38:22822313.

Koltchinskii, V. and Yuan, M. (2010). Sparsity in multiple kernel learning. *The Annals of Statistics*, 38:3660–3695.

Li, T., Zhang, C., and Ogihara, M. (2004). A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20:24292437.

Lin, Y. and Zhang, H. (2006). Component selection and smoothing in multivariate nonparametric regression. *The Annals of Statistics*, 34:22722297.

Long, N., Gianola, D., Rosa, G., and Weigel, K. (2011). Application of support vector regression to genome-assisted prediction of quantitative traits. *Theoretical and Applied Genetics*, 123(7):1065–74.

Meier, L., van de Geer, S., and Buhlmann, P. (2009). High-dimensional additive modeling. *The Annals of Statistics*, 37:3779.

Ravikumar, P., Liu, H., Lafferty, J., and Wasserman, L. (2009). Spam: Sparse additive models. *Journal of the Royal Statistical Society: Series B*, 71:10091030.

Rokach, L. and Maimon, O. (2008). *Data mining with decision trees: theory and applications.* World Scientific Pub Co Inc.

Stone, C. J. (1985). Additive regression and other nonparametric models. *The Annals of Statistics*, 13:689–705.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288.

Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10).

Wood, S. (2003). Thin plate regression splines. *Journal of the Royal Statistical Society: Series B*, 65(1):95–114.

Wood, S. N. (2008). Fast stable direct fitting and smoothness selection for generalized additive models. *Journal of the Royal Statistical Society:Series B*, 70:495–518.

Zhang, H., Wang, H., Chen, M.-s., and Yuan, Z. (2012). Improving accuracy for cancer classification with a new algorithm for genes selection. *BMC Bioinformatics*, 13(298):1471–2105.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 67:301–320.

Appendix A

R Code for MSR-continuous

```
#Input x which is a nxp matrix of n observations with p covariates.
#Input y which is a vector of n observations for the response variable.
#Then run the following:

name=colnames(x)
nm= dim(x)
n=nm[1]
m=nm[2]
betah=matrix(0, nrow=1, ncol=m)
colnames(betah)=colnames(x)
mm=m
vn=dimnames(x)[[2]]
pv=numeric()          #storage for p-values
max.steps=2
u=data.frame(coe=rep(0,m),name)

library(mgcv)
```

```
###compute pvalues
for (j in 1:m) {
n2=n
x2=x[,j]
y2=y
bb=gam(y2~s(x2))
nt=summary(bb)$s.pv
pv[j]=nt
}
Pval=data.frame(cbind(vn,pv))
colnames(Pval)=c("vn","pv")


# Comment - H.Wang
# After calculating p-values for each predictor with the response
# variable, need to discard those variables with p-values large.
# Then further selection is only among the variables kept.
####
aa=2
alfa=0.0005
pvkeep=pv[pv<aa]
xold=x
vnold=vn
mold=m
m=mkeep=sum(pv<aa)
vn=vn[pv<aa]
vlist=seq(mold)[pv<aa]
```

```
x=x[,pv<aa]



beta=seq(0,1,length=max.steps)  # possible values of beta
Pmin=min(pvkeep)
pnew=which.min(pvkeep)   #get index number of the with smallest p-value
enter=NULL
fx=matrix(0,n,mkeep)     #storage for fitted values from smoothing spline
#coeff=numeric()      #storage for coefficients of fx
#coeff[-(1:m)]=0      # those variables with large pvalues should not enter
coeff=NULL
bfxhat=rep(0,n)      #coeff*fx
e=y                  #residuals
plist=NULL

k=1; b=0;
while ( (k<mkeep) & (sum(coeff==0)<2) & (sum(plist)<alfa) ) {
enter[k]=vn[pnew]
###smooth spline
fit=gam(e~s(x[,pnew]))
fx[,k]=drop(fit$fitted.values)



if (m==2) {

      xx=x[,-pnew]
```

```
   pv=NULL
   for (i in max.steps:1) {
    coef=beta[i]
    bfx=bfxhat+coef*fx[,k]
da=data.frame( X=bfx, Y=y, trt=rep(1, n2), A=rep(0, n2)   )
     cc=gam(y~s(bfx))
     pval=summary(cc)$s.pv
     R=ifelse(max(abs(bfx))<1e-4,1, pval)


      res=y-bfx
      x2=xx
      y2=res
      dat=data.frame(X=x2, Y=y2, trt=rep(1, n2), A=rep(0, n2)   )
   NPT=summary(gam(y2~s(x2)))$s.pv
 pvthis=NPT
      pv=c(pv, pvthis) # keep all the pvalues from using all beta values
      b=ifelse(abs(pvthis-R)<alfa,0,abs(pvthis-R))
      plist=c(plist, b)


if (b==0) break()
                        }
 bfxhat=bfx
 e=y-bfxhat
 coeff[mkeep]=ifelse((b==0), 1, 0)  # what is this?
 vn=colnames(x)[-pnew]
 enter[mkeep]=vn
 fit=gam(e~s(xx))
```

```
    fx[,mkeep]=drop(fit$fitted.values)

    coeff[k]=coef

    totfx=bfx+(coeff[mkeep]*fx[,mkeep])

    tres=y-totfx

           } else {

x=x[,-pnew]

nm=dim(x)

n=nm[1]

m=nm[2]

vn=dimnames(x)[[2]]


    for (i in max.steps:1) {

    coef=beta[i]

    bfx=bfxhat+coef*fx[,k]

    da=data.frame(X=bfx, Y=y, trt=rep(1, n2), A=rep(0, n2)   )

    cc=gam(y~s(bfx))

    pval=summary(cc)$s.pv

    R=ifelse(max(abs(bfx))<1e-4,1,pval)



    res=y-bfx

    pv=numeric()

      for (j in 1:m) {

      x2=x[,j]

      y2=res

      dat=data.frame(X=x2, Y=y2, trt=rep(1, n2), A=rep(0, n2)   )

NPT=summary(gam(y2~s(x2)))$s.pv
pv[j]=NPT
                     }
      a=which.min(abs(pv - R))
      b=ifelse(abs(pv[a]-R)<alfa,0,abs(pv[a]-R))
      #cat(i ,"enter",enter[k],coef,R,a,b,"\n")
      plist=c(plist, b)
if (b==0) {break()}
                            }
      pnew=a
      coeff=c(coeff, coef)
      bfxhat=bfx
      e=y-bfxhat

                   }
k=k+1
}

# pick up the last one missed.
d=max(seq(length(coeff))[abs(coeff)>1e-6])+1
if (d<=mkeep  ) {
coeff[d]= 1
lastfit=gam(e~s(xold[,vnold==enter[d]]))
bfxhat=bfxhat+lastfit$fitted.values
err=mean((lastfit$residuals)^2)
} else err=mean(tres^2)

coeff
```

```
enter


xbar=err

v = data.frame(coeff,enter)

updates=v[match(u$name,v$enter),"coeff"]

u$coe=ifelse(!is.na(updates), updates, u$coe)

attach(u)

coe


#Output from this code will consist of

#enter, list of variables selected

#xbar, prediction error

#bfxhat, list of predicted values
```

Appendix B

R Code for Binary Case

B.1 R Code for MSR-binary

```
#Input a dataframe with two arguments, (Y0,X0)
#Y0 is the binary response variable with values either 0 or 1
#X0 is a nxp matrix which consists of p independent variables

dat=data.frame(Y0,X0)

#To run the MSRb algorithm,
MSRb=msr.bin(dat)

#Output from this code will consist of
#zfit, the predicted values for the training data
#ypred, the predicted values for the test data
#pclasif, the proportion of correct classification
#vselect, the variables selected in the jth fold
#nvselect, the number of variables selected in the jth fold
#TPFP1, the number of true positive variables and false
```

```
#positive variables in the jth fold

source("nptest.r")
msr.bin=function(dat, useprior=F,alfa=0.00005, nfold=3,aa=0.5){
set.seed(gmr)
nn=dim(dat)[2]
mm=dim(dat)[1]
nsize=mm
datay=dat[,1]
datax=dat[,2:nn]
loca1=sample((seq(nsize))[datay<0.5])
loca2=sample((seq(nsize))[datay>0.5])
folds1= split(loca1, rep(1:nfold, length = length(loca1)) )
folds2= split(loca2, rep(1:nfold, length = length(loca2)) )

allfolds=mapply(c, folds1, folds2, SIMPLIFY=FALSE)
result=NULL
vselect=NULL
numvselect=NULL
nvselect=NULL
yolds=datay
xolds=datax
library(mgcv)
vselect.fold=list()

for(jj in 1:nfold){
ypos= (1:nsize)[allfolds[[jj]] ]
```

```
yte=yolds[ypos]    # y for the test sample for fold jj

ytr=yolds[-ypos]   # y for the training sample for fold jj

xte=xolds[ypos,]   # x for the test sample for fold jj

xtr=xolds[-ypos,]  # x for the training sample for fold jj

ynew=yte

xnew=xte

y=ytr

x=xtr




name=colnames(x)

nm= dim(x)

n=nm[1]

m=nm[2]

betah=matrix(0, nrow=1, ncol=m)

colnames(betah)=colnames(x)

vn=dimnames(x)[[2]]

pv=numeric()          #storage for p-values

max.steps=2

u=data.frame(coe=rep(0,m),name)



for (j in 1:m) {

x2=x[,j]

y2=y

bb=t.test(x2[y2>0], x2[!(y2>0)])
```

```
nt=bb$p.value
pv[j]=nt

}
Pval=data.frame(cbind(vn,pv))
colnames(Pval)=c("vn","pv")


if (min(pv)>aa) {
psuc=mean(ytr)
pred=rep(ifelse(psuc<0.5,0,1),length(ynew))
enter=NA
numvselect=0
} else {


pvkeep=pv[pv<aa]
xold=x
vnold=vn
mold=m
m=mkeep=sum(pv<aa)
vn=vn[pv<aa]
vlist=seq(mold)[pv<aa]

x=x[,pv<aa]
```

```
enter=NULL
fx=matrix(0,n,mkeep) #storage for fitted values from smoothing spline
coeff=NULL
bfxhat=rep(0,n)
yold=y
e=y
plist=NULL


# variables significant at 0.01 level with Bonferroni correction
# should be included in the final model
if ((sum(pv<0.01/m)>0)&(sum(pv<0.01/m) <n/2)){
 surekeepindex=seq(mold)[pv<0.01/m]
 enter=vn[ surekeepindex]
 k= length(surekeepindex)
 coeff=rep(1, k)
 fitpred=fit.and.predictGAM(x,e, x,enter, coeff)
 e=fitpred$e; bfxhat=fitpred$bfhat
 x=x[,!(pvkeep<0.01/m)]
 vn=vn[!(pvkeep<0.01/m)]
 vlist=seq(m)[!(pvkeep<0.01/m)]
 pvkeep=pvkeep[!(pvkeep<0.01/m)]
 m=ncol(x)
} else {enter=NULL; k=0}

beta=seq(0,1,length=max.steps)  # possible values of beta
Pmin=min(pvkeep)
```

```
pnew=which.min(pvkeep) #get index number of the variable with smallest p-value

b=0;
stopcond =T
while ( (k<mkeep) & (sum(coeff==0)<2) &  stopcond) {

k=k+1



if (k==1) {

x2new=x[,pnew]
fitq=gam(e~s(x2new),family=binomial(link = "logit"))
fx[,k]=predict(fitq)
          } else {
x2new=x[,pnew]
ph=1/(1+exp(-bfxhat))
wt=ph*(1-ph)
if (max(abs(wt))<10^(-5)) wt=rep(1,length(ph))
fitq=gam(e~s(x2new),weights=wt)
fx[,k]=predict(fitq)   #drop(fit$fitted.values)
     }
if (m==2) {
 pv=NULL
 for (i in max.steps:1) {
 coef=beta[i]
 bfx=bfxhat+coef*fx[,k]
 dd=gam(yold~bfx,family=binomial(link = "logit"))
 phat=1/(1+exp(-bfx))
 w=phat*(1-phat)
 if (max(abs(w))<10^(-5)) w=rep(1,length(phat))
 pval= summary(dd)$p.pv[2]
 R=ifelse(max(abs(bfx))<1e-4,1, pval)
 res=residuals(dd,type="working")
 x2=xx
 y2=res
 daty21=data.frame(X=x2, Y=y2, trt=rep(1, length(y2)), A=rep(0, length(y2))   )
 NPT=NPtest.indept.alt( daty21, k=3, estpower=T, eta.simple=0, alpha=0.01)
 pvthis=NPT$pvalue.sim
 pv=c(pv, pvthis) # keep all the pvalues from using all beta values
 b=ifelse(abs(pvthis-R)<alfa,0,abs(pvthis-R))
 plist=c(plist, b)
if (b==0) {
bfxhat=bfx
e=residuals(dd,type="working")
coeff[mkeep]=ifelse((b==0), 1, 0)
vn=colnames(x)[-pnew]
enter[mkeep]=vn
break()
}
}

ph=1/(1+exp(-bfxhat))
wt=ph*(1-ph)
```

```
if (max(abs(wt))<10^(-5)) wt=rep(1,length(ph))
    fit=gam(e~s(xx),weights=wt)
    fx[,mkeep]=drop(fit$fitted.values)
    coeff[k]=coef
    totfx=bfx+(coeff[mkeep]*fx[,mkeep])
    tres=residuals(fit,type="working")
           } else {
 if (k==1) {enter[k]=vn[pnew]; x=x[,-pnew]; coeff=1; k=k+1}
nm=dim(x)
n=nm[1]
m=nm[2]
vn=dimnames(x)[[2]]

for (i in max.steps:1) {
coef=beta[i]
bfx=bfxhat+coef*fx[,k]
dd=gam(yold~bfx,family=binomial(link = "logit")) #to get p_c
phat=1/(1+exp(-bfx))
w=phat*(1-phat)
if (max(abs(w))<10^(-5)) w=rep(1,length(phat))
pval=summary(dd)$p.pv[2] #this is the p_c
R=ifelse(max(abs(bfx))<1e-4,1,pval)

res=residuals(dd,type="working")   #z-bfx
pv=numeric()
for (j in 1:m) {
x2j=x[,j]
y2=res
daty2=data.frame(X=x2j, Y=y2, trt=rep(1, length(y2)),A=rep(0,length(y2)))
NPT=NPtest.indept.alt( daty2, k=3, estpower=T, eta.simple=0, alpha=0.01)
pv[j]=NPT$pvalue.sim
                      }
a=which.min(abs(pv - R))
b=ifelse(abs(pv[a]-R)<alfa,0,abs(pv[a]-R))
plist=c(plist, b)
if (b==0) {
  pnew=a
  enter[k]=vn[pnew]
  x=x[,-pnew]
  cat("break", vn[pnew], "\n")
  if (coef==0) {
  coeff=c(coeff, 10)
  fitpred1=fit.and.predictGAM(xold,yold, xold,enter,coeff=rep(1,length(enter)))
    bfxhat=fitpred1$ypred
e= residuals(fitpred1,type="working")
vn=colnames(x)[-pnew]
} else{
    coeff=c(coeff, coef)
    bfxhat=bfx
    e=residuals(dd,type="working") #z-bfxhat
vn=colnames(x)[-pnew]
}
break()
  }
```

```
                    }
   }
stopcond=(min(plist[ length(plist):(length(plist)-1)] )<alfa)
}

enter=enter[coeff>0]



#PREDICTION PART


fitpred=fit.and.predictGAM(xold,yold, xte,enter, coeff)
ypred=fitpred$ypred  # predict at link scale
enter
numvselect=length(enter)
pred=1/(1+exp(-ypred))
if (useprior==T) psuccess=sum(yold)/length(yold) else psuccess=0.5
pred[pred<psuccess]=0
pred[pred>=psuccess]=1
}
resnew=cbind(yte,pred)
result=rbind(result,resnew)
vselect.fold[[jj]] =enter
vselect=c(vselect,enter)
nvselect=c(nvselect,numvselect)
}
nclasif=sum(result[,1]==result[,2])
pclasif=nclasif/nsize
```

```
meanvs=mean(nvselect)

tpfpmsr=true.postiveMsr1(vselect.fold, nvselect,nfold=length(vselect.fold))

list(nclasif=nclasif,pclasif=pclasif, vselect=vselect, nvselect=nvselect,

vselect.fold=vselect.fold,predicted=result,TPFP1=unlist(tpfpmsr))

}



fit.and.predictGAM=function(xold,yold, xte,enter,coeff=rep(1,length(enter)) ){

xfinal=xold[,enter]

xnew=xte[,enter]

if (sum(coeff)==1) {

tt=length(xnew)

g=rep(0,tt)

cres=yold

ypred=rep(0,tt)

z=xfinal

h=gam(cres~s(z),family=binomial(link = "logit"))

r=residuals(h,type="working")

newd=data.frame(z=xnew)

bf=predict(h)

g=predict(h,newd)

cres=r

ypred=ypred+g

row.names(ypred)=NULL

zfit=bf+cres


} else {
```

```
tt=dim(xnew)[1]

p=ncol(xfinal)

g=matrix(data=0,nrow=tt,ncol=p)

cres=yold

ypred=rep(0,tt)

for (w in 1:p){

if (w==1) {

z=xfinal[,w]

h=gam(cres~s(z),family=binomial(link = "logit"))

r=residuals(h,type="working")

newd=data.frame(z=xnew[,w])

bf=predict(h)

g[,w]=predict(h,newd)

cres=r

ypred=ypred+g[,w]

zfit=bf+cres} else {

z=xfinal[,w]

ptt=1/(1+exp(-bf))

wtt=ptt*(1-ptt)

if (max(abs(wtt))<10^(-5)) wtt=rep(1,length(ptt))

h=gam(r~s(z),weigths=wtt)

fits=h$fitted.values

bf=bf+fits

r=residuals(h,type="working")

newd=data.frame(z=xnew[,w])

g[,w]=predict(h,newd)

cres=r
ypred=ypred+g[,w]
zfit=bf+cres}
     } }
list(zfit=zfit, ypred=ypred, bfhat=bf, e=r)
}
```

B.2 R code for GLMNET

```
#Input a dataframe with two arguments, (Y0,X0)
#Y0 is the binary response variable with values either 0 or 1
#X0 is a nxp matrix which consists of p independent variables

dat=data.frame(Y0,X0)

#To run the MSRb algorithm,
glmnet=glmnetBin(dat)

#Output from this code will consist of
#ypred, the predicted values for the test data
#pclasif, the proportion of correct classification
#vselect, the variables selected in the jth fold
#numvar, the number of variables selected in the jth fold
#TPFP1, the number of true positive variables and false
#positive variables in the jth fold

glmnetBin=function(dat){
```

```
library(glmnet)

set.seed(gmr)

nfold=3

nn=dim(dat)[2]

mm=dim(dat)[1]

nsize=mm

y=dat[,1]

x=dat[,2:nn]

n=length(y)

loca1=sample((seq(n))[y<0.5])

loca2=sample((seq(n))[y>0.5])

folds1= split(loca1, rep(1:nfold, length = length(loca1)) )

folds2= split(loca2, rep(1:nfold, length = length(loca2)) )


allfolds=mapply(c, folds1, folds2, SIMPLIFY=FALSE)


yolds=y

xolds=x

numvar=NULL

resultgnet=NULL

vselect.fold=list()

for(jj in 1:nfold){

ypos= (1:nsize)[allfolds[[jj]] ]

yte=yolds[ypos]    # y for the test sample for fold jj

ytr=yolds[-ypos]   # y for the training sample for fold jj

xte=xolds[ypos,]   # x for the test sample for fold jj

xtr=xolds[-ypos,]  # x for the training sample for fold jj
ynew=yte
xnew=as.matrix(xte)
y=unlist(ytr)
x=as.matrix(xtr)
cvob1=cv.glmnet(x,y,family="binomial")
glmnetfit=glmnet(x,y,family="binomial", lambda=cvob1$lambda.min)
pall=length(glmnetfit$beta)
vselect.fold[[jj]]=(seq(pall))[abs(as.vector(glmnetfit$beta))>1e-5 ]
tw=predict(cvob1,xnew,type="response",s="lambda.min")
predg=tw
predg[predg<0.5]=0
predg[predg>=0.5]=1
resgnet=cbind(yte,predg)
resultgnet=rbind(resultgnet,resgnet)
varn=cvob1$nzero[(cvob1$lambda==cvob1$lambda.min)]
numvar=c(numvar,varn)
}
print(resultgnet)
nclasif=sum(resultgnet[,1]==resultgnet[,2])
clasifg=sum(resultgnet[,1]==resultgnet[,2])/nsize
tpfpglm=true.postive1(vselect.fold, nfold=length(vselect.fold))
list(numvar=numvar, nclasif=nclasif, pclasif=clasifg,
vselect.fold=vselect.fold,TPFP1=unlist(tpfpglm))
}
```

B.3 R Code for True and False Positive

```
true.postive= function(vselect.fold, nfold=length(vselect.fold)){
```

```
tpfp=NULL
for (i in 1:nfold){
tp=sum((vselect.fold[[i]] ==1) | (vselect.fold[[i]] ==2) )
fp= sum(vselect.fold[[i]] >2)
tpfp=rbind(tpfp, c(tp,fp) )
}
colnames(tpfp)=c("tp", "fp");
row.names(tpfp)=paste("fold",1:nfold, sep="")
tpfp
}

true.postive1= function(vselect.fold, nfold=length(vselect.fold)){
tpfp=NULL
for (i in 1:nfold){
tp=sum((vselect.fold[[i]] ==1) )
fp= sum(vselect.fold[[i]] >1)
tpfp=rbind(tpfp, c(tp,fp) )
}
colnames(tpfp)=c("tp", "fp");
row.names(tpfp)=paste("fold",1:nfold, sep="")
tpfp
}



true.postiveMsr1= function(vselect.fold, nvselect,nfold=length(vselect.fold)){
tpfp=NULL
```

```
for (i in 1:nfold){
tp=sum((vselect.fold[[i]] =="X1")  )
fp= nvselect[i] -tp
tpfp=rbind(tpfp, c(tp,fp) )
}
colnames(tpfp)=c("tp", "fp");
row.names(tpfp)=paste("fold",1:nfold, sep="")
tpfp
}


true.postiveMsr2= function(vselect.fold,nvselect,nfold=length(vselect.fold)){
tpfp=NULL
for (i in 1:nfold){
tp=sum((vselect.fold[[i]] =="X1") | (vselect.fold[[i]] =="X2")   )
fp= nvselect[i] -tp
tpfp=rbind(tpfp, c(tp,fp) )
}
colnames(tpfp)=c("tp", "fp");
row.names(tpfp)=paste("fold",1:nfold, sep="")
tpfp
}




true.postiveMsr4= function(vselect.fold, nfold=length(vselect.fold)){
tpfp=NULL
```

```
for (i in 1:nfold){
tp=sum((vselect.fold[[i]] =="X1") | (vselect.fold[[i]] =="X2")  |
(vselect.fold[[i]] =="X3")  | (vselect.fold[[i]] =="X4") )
fp= length(vselect.fold[[i]]) -tp
tpfp=rbind(tpfp, c(tp,fp) )
}
colnames(tpfp)=c("tp", "fp");
row.names(tpfp)=paste("fold",1:nfold, sep="")
tpfp
}
```

B.4 R Code for NPtest

```
################################################################################

# i1 is the i1 th group; n is the vector of sample sizes;
# position.i1 function gives the starting and end position of covariate
#values in the i1th group among the vector listing all covariate values.
# e.g., covariate values in group 1 start from 1st value to the n1 th value;
#those in group 2 start from n1+1 and end at n1+n2 th value.

position.i1=function(i1, n){
if (i1==1) lower=1 else lower=sum(n[1:(i1-1)])+1
upper=sum(n[1:i1])
c(lower, upper)
}
```

```
############# map the index over r=1,...,N to i=1, ...a, j=1, ..., n_i

# r is an integer; n is a vector of the sample sizes


mapindex=function(r, n){

aaa=length(n)

sumn=numeric()

for ( i in 1:aaa)  sumn=c(sumn, sum(n[seq(i)]) )

imap=sum(sumn<r)+1

jmap=r-sum(n[ seq(aaa)[(sumn<r)]] )

c(imap, jmap)

}





makepseudo=function(N,n, k, a, alltrt){

psudo<-array(0, c(a, sum(n), k))

index<-array(0, c(a,sum(n), k))


 ### Augment observations for each cell

##**************************************



for (i in 1:a){


for (j in 1:N){
```

```
if (i==1){



if ( j<= n[1] ) {
newtrt<-alltrt[,1:n[1]]
total<-ncol(newtrt)
jj<-j
}

if  (j>=n[1]+1) {
 newtrt<-cbind(alltrt[,1:n[1]], alltrt[, j])
total<-jj<- ncol(newtrt)
}
}



if (i>1) {

if ((j<=sum(n[1:i]))& (j>=sum(n[1:(i-1)])+1) ) {
newtrt<-alltrt[,(sum(n[1:(i-1)])+1): sum(n[1:i])]
total<- ncol(newtrt)
jj<- j-sum(n[1:(i-1)])
} else {
newtrt<-cbind(alltrt[,(sum(n[1:(i-1)])+1): sum(n[1:i])], alltrt[,j] )
total<-jj<-ncol(newtrt)
}
}



newtrt[3, ]<-rank(newtrt[2, ])
flag<-((jj==total)& (jj>n[i])& c(rep(T, total-1), F)  )  | (jj<=n[i])
if ((jj==total) & (jj>n[i]) ) {
              newtrt[3, -jj]<- rank(newtrt[2, -jj])
              total<-total-1
              }
target<-newtrt[3, jj ]
newtrt<-newtrt[, flag]



if (trunc(target) <= ((k-1)/2) )
 {psudo[i,j,  ]<- newtrt[1, order(newtrt[3, ])[1:k]]
 index[i,j,  ]<- seq(1, total)[ order(newtrt[3, ])[1:k]]
 }
if (trunc(target) > (total- ((k-1)/2)))
 {psudo[i,j,  ]<- newtrt[1, order(total-newtrt[3, ])[1:k] ]
 index[i,j,  ]<- seq(1, total)[ order(total-newtrt[3, ])[1:k]]
 }
if ((trunc(target)  <=(total-(k-1)/2 ) ) & (trunc(target) >((k-1)/2) )   )
  {
 psudo[i,j,  ]<-   newtrt[1, order((abs(newtrt[3,]-trunc(target) ) )) [1:k] ]
 index[i,j, ]<- seq(1, total)[ order((abs(newtrt[3,]-trunc(target) ) )) [1:k]]
  }
```

```
#cat(j, "done\n")
 }  #end of j







}   #end of i


list(psudo=psudo, index=index)
}






NPtest.indept.alt= function( dat, k=3, estpower=T, eta.simple=0, alpha=0.01)
{
X=dat$X;  trt=dat$trt; Y=dat$Y;  #Y=unlist(tapply(dat$Y, trt, standard) )
ranksuse=unlist(tapply(X, trt, rank) )  ### replace with rank(X)
alltrt=rbind(Y, X, ranksuse )

n=unlist(tapply(rep(1, nrow(dat)), trt, sum))  ### replace with length(X)
N=sum(n); a=length(n)
#if (estpower==T) alltrt2=rbind(dat$A, X, ranksuse ) else
#alltrt2=rbind(rep(0, N), X,  ranksuse)
   ## put the data in increasing order of X (within each trt)
for (i1 in 1:a){
```

```
locationi1=position.i1(i1,n);
orderwant=order(alltrt[2,locationi1[1]:locationi1[2]])+locationi1[1]-1;
alltrt[,locationi1[1]:locationi1[2] ]=  alltrt[,orderwant]
#  alltrt2[,locationi1[1]:locationi1[2] ]=  alltrt2[,orderwant]
    }

psudodat=makepseudo(N,n, k, a, alltrt)
#psudoA=makepseudo(N,n, k, a, alltrt2)
psudo=psudodat$psudo; index=psudodat$index
#psudo2=psudoA$psudo



##*****************************************
#cellmeanA<-apply(psudo2, c(1,2), mean)
#meanrkA=apply(psudo2,1, mean)
#asy.mean.alt=k*sum( (cellmeanA-matrix(rep(meanrkA,N),
#ncol=N)^2)/((sum(n)-1)*a)# asymptotic mean under alt
##*****************************************

cellmean<-apply(psudo, c(1,2), mean)
# apply(cellmean, 1, var)
colmean=apply(psudo, 2, mean)
sig<- cov(t(cellmean))  # diagonal part gives the \hat\sigma_{1,i}^2
                        # and off-diagonal part gives \hat\sigma_{1,i_1, i_2}
## calculate all $\widehat{\sigma}_i^2(X_{ij})$
sigXij<-apply(psudo, c(1, 2), var)
      # get a axN matrix with \hat\sigma_i^2(X_{ij}) =sigXij[i, j]
```

```
#sigXij<-apply(makepseudo(N,n,4*k,a,alltrt),c(1, 2), var)
# use 4k pseudo obs for var approximate
#sigXij=t(apply(sigXij,1,function(x) smooth.spline(x)$y))
# use smoothed version to approximate var
#MSTd=k*a*sum( (colmean-mean(psudo)  )^2 )  /(sum(n)-1)

# calculate $\overline{U}_{i..}$
meanrk=apply(psudo,1, mean)

# calculate $B_N$
MSTphi=k*sum( (cellmean-matrix(rep(meanrk,N),ncol=N))^2  )  /((sum(n)-1)*a)

#MSTc=k*sum((cellmean-matrix(rep(meanrk,N),ncol=N)-matrix(rep(colmean,a),ncol=N,
 byrow=T)+mean(cellmean) )^2  )/((a-1)*(N-1))

# calculate MSE i.e. $W_N$
MSE=  sum((psudo-array(rep(cellmean, k), c(a, sum(n), k)))^2)/(sum(n)*a*(k-1))

#Tsinter=(sqrt(sum(n)) * (MSTc-MSE))
#Tsc=(sqrt(sum(n)) * (MSTd-MSE))
Tss=(sqrt(sum(n)) * (MSTphi-MSE))

## calculate Td1 and Td2 for diagnostics
#offdiagsum=function(x) sum(matrix(x)%*%
#matrix(x, ncol=length(x)))-sum(x^2) #\sum_{m\ne m'} x_m x_{m'}
#Td1=k*mean( apply(cellmean, 2, offdiagsum) )/a
```

```
#Td2=mean(apply(psudo, c(1,2), offdiagsum ) )/(k-1)

##**************************************

#Calculate estimate of variance for test statistics

# count is a matrix;first 3 columns give the value of i1 j2 i;the last column
# gives the number of times X_{ij_2} is used in construction of windows for
# all covariate values in group i_1

count<-matrix(-1, a^2*N, 4)
whereini=0
for( i1 in 1:a){
for (j2 in 1:N){
for (i in 1:a){
  whereini=whereini+1
  if (i1==1) lower=1 else lower=sum(n[1:(i1-1)])+1
  upper=sum(n[1:i1])
  counti1j2i=sum(index[i,lower:upper, ]==(   (mapindex(j2, n)[1]==i
  ) *mapindex(j2, n)[2]) )  ## this is the line different from NP.test.old
    count[whereini, ]=c(i1, j2, i, counti1j2i)
}}}



           tau3=0
for (jp in 2:n[i]){
for (j in (max(1, (jp-k+1)): (jp-1)) ){
```

```
tau3=tau3+ (  (k-jp+j)^2 + (k-jp+j)-2*(jp-j<=((k-1)/2) )    )*(jp-j<=k-1
              ) * sigXij[j] *sigXij[jp] *(j!=jp)
} }
tau3=tau3*4/(sum(n)*(k-1)^2)



tauAsys=tau3
pvalue.sim=1-pnorm(Tss/sqrt(tauAsys))
power2=pnorm(qnorm(alpha, mean=0, sd=sqrt(tauAsys), lower.tail=F ),
    mean=eta.simple*k/a,   sd=sqrt(tauAsys), lower.tail=F)
list(tauAsys=tauAsys,Tss=Tss,pvalue.sim=pvalue.sim,  power2=power2)
}



#NPtest.indept.alt( dat, k=3, estpower=T, eta.simple=0, alpha=0.01)
```