UNIVERSITY OF CALIFORNIA
RIVERSIDE

Essays on Classification, Variable Selection and Statistical Inference

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Economics

by

Jianghao Chu

June 2019

Dissertation Committee:

Professor Aman Ullah, Co-Chairperson
Professor Tae-Hwy Lee, Co-Chairperson
Professor Gloria Gonzalez-Rivera

The Dissertation of Jianghao Chu is approved:

Committee Co-Chairperson

Committee Co-Chairperson

University of California, Riverside

Acknowledgments

I would like to express my deepest gratitude to my advisors, without whose help, I would not have been here. In particular, I appreciate Professor Aman Ullah's wise guidance, patience, motivation and immense knowledge leading me on the track. The research skills I have learned from him will definitely be my lifetime asset. I enjoy every efficient meeting with Professor Tae-Hwy Lee, as his insightful comments are inspiring and save me a great amount of time. I cannot thank them enough as they have always been supporting my decisions and providing me their advice without any reservation. They have shown me, by their endless passion for research and selfless devotion to students, what a scholar should be. I am also greatly indebted to Professor Gonzalez-Rivera for her helpful advice and feedback. I would never be able to finish my dissertation without the support and encouragement of all my committee members.

I would like to thank the entire economics department for providing me with an ideal research environment. I am blessed to have become a member of the economics department at the University of California, Riverside. I am equipped with a sharper and broader vision in research by talking to and receiving feedback from our professors such as Professor Urmee Khan and Professor Ruoyao Shi. I would like to give special thanks to Professor Steven Helfand for his support to the Graduate Student Association and Gary Kuzas for his help in dealing with university regulations which have saved me countless times from breaking down.

I would like to thank my friend and roommate, Yun Luo, for helping me with numerous issues big and small to get used to the life in the United States. I would like to thank my friend Hanbyul Ryu and his family for inviting me for dinner and spending time with me when I am down. I would also like to thank my friends, Hao

Xu, Mingyuan Jia, Seolah Kim, and Ran Wang for their help in every stage of my academic journey.

Last but not least, I would like to thank my parents, Houxu Chu and Yiqin Jiang, who have supported all my decisions unconditionally and are always there for me through good and bad times.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

Essays on Classification, Variable Selection and Statistical Inference

by

Jianghao Chu

Doctor of Philosophy, Graduate Program in Economics
University of California, Riverside, June 2019
Professor Aman Ullah, Co-Chairperson
Professor Tae-Hwy Lee, Co-Chairperson

This dissertation covers topics in classification with high-dimensional data, variable selection in sparse semiparametric single-index models and statistical inference under heteroskedasticity. In particular, Chapter 1 provides the motivation and background of the dissertation.

Chapter 2 provides a summary of boosting methods for classification, namely Discreet AdaBoost, Real AdaBoost, P-AdaBoost, Gentle AdaBoost and LogitBoost. We compare these methods with alternative machine learning classification tools such as Deep Neural Network and demonstrate the empirical applications in economics, such as prediction of business cycle turning points and directional prediction of stock price indexes.

Chapter 3 generalizes the Discreet AdaBoost shown in Chapter 2 for binary classification problem with state-dependent loss functions. We introduce Asymmetric AdaBoost that solves the asymmetric maximum score problem with high-dimensional data. Asymmetric AdaBoost produces a nonparametric classifier via minimizing the "asymmetric exponential risk" which is a convex surrogate of the traditional non-convex score risk or 0-1 risk. The convex risk function gives huge computation

advantage over non-convex risk functions, e.g. Maximum Score (Manski, 1975, 1985), especially when the data is high-dimensional.

Chapter 4 considers the "Regularization of Derivative Expectation Operator" (RODEO) of Lafferty and Wasserman (2008) and propose a modified RODEO algorithm for sparse semiparametric single-index models which we call the SIM-RODEO. The SIM-RODEO method is able to distinguish relevant explanatory variables from irrelevant variables and gives a competitive estimator for the model. In addition, the algorithm finishes in a reasonable period of time.

Chapter 5 investigates the methods for statistical inference under the presence of heteroskedasticity of unknown form in the disturbances of linear regression models. We propose a F-type (t^2-type) test statistic for testing regression parameters under the heteroskedasticity of unknown form. The accuracies of the test statistic are confirmed by extensive Monte Carlo experiments. And Chapter 6 concludes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many important variables in economics, as well as other disciplines, are binary. For example, whether the economy is going into an expansion or a recession, whether the stock market is going up or going down, whether the Federal Reserve Bank of the United States of America should increase the interest rate, whether a mortgage applicant will default in the future, and etc.

Let

$$\pi(x) \equiv P(y = 1|x)$$

and y takes value 1 with probability $\pi(x)$ and -1 with probability $1 - \pi(x)$. The studies on making the best forecast on y can be classified into two classes: point prediction and probability prediction (Lahiri and Yang, 2012). Probability prediction is focusing on estimating the right probability model $\hat{\pi}(x)$ such as the Logit and Probit models (Gaddum, 1933; Bliss, 1934a,b), then making the forecast with $\hat{\pi}(x) > 0.5$ using the estimated probability model. Point prediction is determined to get the optimal forecast rule $\hat{\pi}(x) > 0.5$ without having to (correctly) model the probability of the events such as the maximum score approach (Manski, 1975, 1985).

Given the availability of high-dimensional data, both methods have been improved to incorporate a large number of independent variables (x). Friedman et al. (2010a) introduce regularized logistic regression which adds an L_1 penalty on the coefficients of independent covariates and solve the problem of excessive independent covariates by shrinking the coefficients of unimportant covariates to zero. Freund and Schapire (1996) introduce the AdaBoost algorithm which uses a functional descent procedure and selects the independent variables sequentially instead of all at once.

Both methods are widely used and perform well in practice. However, both methods take 0.5 as the cutoff. The focus of forecasting in statistics and computer science is often on having a larger probability of forecasting y correctly. That goal requires the usage of a symmetric loss function. In other words, wrong predictions are given the same penalty regardless of whether it is a false positive (type I error) or false negative (type II error) prediction. Economists, on the other hand, have long been relating the forecasting problem with decision theory which aims at maximizing the utility of the economic agent. They find that people prefer to making more costless mistakes than making a costly one. For example, people are more willingly to arrive at the airport early than late (Granger, 1999). People are more willingly to overestimate the peak water of a dam than to underestimate it (Zellner, 1986). Thus, the optimal decision rule for an economic agent should not be the one that gives the most correct forecasts but the one that helps to prevent making costly mistakes.

Such incentives promote the use of an asymmetric loss function in the estimation process of the optimal forecast which is related to the economic agent's utility function and gives a higher penalty on costly mistakes and lower penalty on costless mistakes. Lee and Yang (2006); Elliott and Lieli (2013); Lahiri and Yang (2012) study the classification problems under asymmetric loss functions. Elliott and Lieli (2013) propose a utility based classifier called maximum utility estimator by using the

maximum score approach of Manski (1985) but include utilities that result in an asymmetric loss function.

This dissertation studies both point prediction methods and probability prediction methods for binary classification in the era of big data. The goal of this dissertation is to propose methods for binary classification that are both analytically consistent and numerically feasible with big data. In addition, I investigate both kinds of methods under the setting of high-dimensionality and/or sparsity which are common setups for big data. However, it is worth to note that the methods proposed in this dissertation would also work under traditional settings where these conditions are ignored. Moreover, this dissertation also investigates the methods for statistical inference under the occurrence of heteroskedasticity which is inevitable when the dependent variable is binary. The rest of the dissertation is organized as follows.

Chapter 2 (joint work with Professor Tae-Hwy Lee and Professor Aman Ullah) compares the performance of different Boosting algorithms, namely Discreet AdaBoost, Real AdaBoost, P-AdaBoost, Gentle AdaBoost and LogitBoost, as well as other machine learning methods, such as the Deep Neural Network, for binary classification with high-dimensional data, i.e. the number of variables in the data exceeds the number of observations. We summarize the above Boosting algorithms in terms of their choices of loss functions, step sizes, and weak learners. Different choices in those specifications lead to different empirical convergence rates and different rates of over-fitting. We compare the Boosting algorithms together with the Deep Neural Network, which is extremely popular and powerful in industrial applications, by simulations to investigate their properties in terms of algorithmic convergence and over-fitting. We also provide comparisons with empirical applications in economics, such as prediction of business cycle turning points and directional prediction of stock price indexes.

Chapter 3 (joint work with Professor Tae-Hwy Lee and Professor Aman Ullah)

focuses on binary classifications with high-dimensional data under state-dependent loss functions. More specifically, we generalize the Discreet AdaBoost shown in Chapter 2 for binary classification problems with state-dependent loss functions. We introduce a convex surrogate loss function of the traditional non-convex loss function used in the maximum score approach Manski (1975, 1985) which we call the "asymmetric exponential risk". We show that minimizing the convex surrogate loss function would give us a classifier that also minimizes the traditional non-convex loss functions. Thus, minimizing the convex surrogate loss function solves the maximum score problem. We also propose a numerical algorithm that minimizes the "asymmetric exponential loss" by building a tree-based nonparametric classifier. The algorithm does not require any parametric assumption and is able to work with high-dimensional data. The resulting nonparametric classifier is more robust than parametric classifiers whose performance depend on the correct specification of the model. In addition, the use of the convex "asymmetric exponential loss" gives huge computation advantage over non-convex risk functions, e.g. Maximum Score (Manski, 1975, 1985). Hence, the proposed algorithm is numerically much more efficient, especially with big data. We show that the risk of the classifier that Asymmetric AdaBoost produces approaches the Bayes risk which is the infimum risk can be achieved by all classifiers. Monte Carlo experiments show that Asymmetric AdaBoost performs better than the commonly used LASSO-regularized logistic regression when the parametric assumption is violated and the sample size is large.

Chapter 4 (joint work with Professor Tae-Hwy Lee and Professor Aman Ullah) considers the problem of probability prediction of binary variables. We extend the "Regularization of Derivative Expectation Operator" (RODEO) of Lafferty and Wasserman (2008a) and propose a modified RODEO algorithm for semiparametric single-index models with many regressors, some of which may be irrelevant to the variable of

interest. We start with the semiparametric single-index model of Ichimura (1993). The semiparametric method is known to be vulnerable to the curse of dimensionality. Hence, it is unable to provide the estimates in a reasonable amount of time with a large number of variables, even if many of them are known to be irrelevant. We take advantage of the sparse nature of the data and select only the relevant variables to reduce the dimensionality of the data and reduce the curse of dimensionality. The algorithm uses a greedy procedure to estimate the semiparametric single-index model (SIM) of Ichimura (1993), all coefficients of the regressors are initially set to start from near zero, then we test iteratively if the derivative of the regression function estimator with respect to each coefficient is significantly different from zero. The basic idea of the modified RODEO algorithm for SIM (to be called SIM-RODEO) is to view the local bandwidth selection as a variable selection scheme which amplifies the coefficients for relevant variables while keeping the coefficients of irrelevant variables relatively small or at the initial starting values near zero. For sparse semiparametric single-index models, the SIM-RODEO algorithm is shown to attain both the consistency in variable selection and the consistency in the estimation of the regression function. In addition, the algorithm is fast to finish the greedy steps. We compare SIM-RODEO with SIM-LASSO method in Zeng et al. (2012). We show that the proposed SIM-RODEO method is consistent for the variable selection and our simulation results show that it has smaller integrated mean squared errors than SIM-LASSO.

Chapter 5 (joint work with Professor Tae-Hwy Lee and Professor Aman Ullah) studies the effect of heteroskedasticity which inevitably appears in problems with limited dependent variables. In particular, we investigate the methods of statistical inference under the presence of heteroskedasticity of unknown form in the disturbances of a linear regression model. Under heteroskedasticity, the ordinary least squares (OLS) estimators of the regression parameters are well known to be consistent.

However, the usual estimators for the covariance matrix of the regression parameters are inconsistent and/or biased. We study the performance of the t-statistic and the F-statistic when a sandwich like heteroskedasticity consistent (HC) variance estimator, e.g. the "White variance estimator", is used. We show that the t-statistic and the F-statistic are ill-behaved in terms of the size of the tests when the heteroskedasticity consistent variance estimators are used. Hence, they tend to over-reject and lead to false discoveries. Recent studies have also pointed out that the over-rejection problem remains problematic in considerably large samples when the data are leveraged/unbalanced. We propose a F-type (t^2-type) test statistic for testing regression parameters under the heteroskedasticity of unknown form. This test statistic is valid for a single linear restriction on the regression parameter including the test for the zero restriction on each coefficient. It is shown that the proposed F-type test statistic can be expressed as a ratio of quadratic forms, and therefore its exact cumulative distribution under the null hypothesis can be easily written, and straightforwardly implemented from the result of Imhof (1961) on the distribution of quadratic form. We also generalize our proposed method as well as other methods for exact inference under heteroskedasticity to deal with both heteroskedasticity and auto-correlation. In addition, we also consider the case of the cross-sectional correlation between individuals in panel models. A numerical calculation of the proposed test statistic, using Imhof (1961), is carried out to present the critical values and probability of rejections under various covariance estimators of regression estimators. The accuracies are confirmed from their corresponding simulation-based results. Chapter 6 is the conclusions.

Chapter 2

Boosting Algorithms for High-dimensional Binary Classification and Class Probability Prediction

2.1 Introduction

A large number of important variables in economics are binary. Let

$$\pi(x) \equiv P(y = 1|x)$$

and y takes value 1 with probability $\pi(x)$ and -1 with probability $1 - \pi(x)$. The studies on making the best forecast on y can be classified into two classes (Lahiri and Yang, 2012). One is focusing on getting the right probability model $\hat{\pi}(x)$, e.g., Logit and Probit models (Bliss, 1934a; Cox, 1958; Walker and Duncan, 1967), then making

the forecast on y with $\hat{\pi}(x) > 0.5$ using the estimated probability model. The other is to get the optimal forecast rule on y directly, e.g., the maximum score approach (Manski, 1975, 1985; Elliott and Lieli, 2013), without having to (correctly) model the probability $\hat{\pi}(x)$.

Given the availability of high-dimensional data, the binary classification or binary probability prediction problems can be improved by incorporating a large number of covariates (x). A number of new methods are proposed to take advantage of the great number of covariates. Freund and Schapire (1997) introduce machine learning method called Discrete AdaBoost algorithm, which takes a functional descent procedure and selects the covariates (or predictors) sequentially. Friedman et al. (2000a) show that AdaBoost can be understood as a regularized Logistic Regression, which selects the covariates one-at-a-time. The influential paper also discusses several extensions to the original idea of Discrete AdaBoost and proposes new Boosting methods, namely Real AdaBoost, LogitBoost and Gentle Boost, which uses the exponential loss or Bernoulli log-likelihood as fitting criteria. Later on, Friedman (2001) generalize the idea to any fitting criteria and proposes the Gradient Boosting Machine. Bühlmann and Yu (2003) and Bühlmann (2006) propose the L_2 Boost and prove its consistency for regression and classification. Mease et al. (2007) use the logistic function to convert the class label output of boosting algorithms into probability and/or quantile predictions. Chu et al. (2018a) show the linkage between the Discrete AdaBoost and the maximum score approach and propose Asymmetric AdaBoost for utility based high-dimensional binary classification.

On the other hand, efforts are made to incorporate traditional binary classification and probability prediction methods into the high-dimensional sparse matrix set-up. The key feature of high-dimensional data is the redundancy of covariates in the data. Hence, methods are proposed to select useful covariates while/before estimation of the

models. Tibshirani (1996) proposes the LASSO that is to add L_1 penalty to including more covariates in the model. Zou (2006) drives a necessary condition for consistency of the LASSO variable selection and proposes the Adaptive LASSO which is showed to enjoy oracle property. LASSO type methods are often used with parametric models such as linear model or logistic model. To relax the parametric assumptions, Lafferty and Wasserman (2008a) propose the Regularization of the Derivative Expectation Operator (RODEO) for variable selection in kernel regression. Chu et al. (2018b) proposes SIM-RODEO for variable selection in semiparametric single-index model. See Su and Zhang (2014) for a thorough review of variable selection in nonparametric and semiparametric models.

This chapter gives an overview of recently developed machine learning methods, namely AdaBoost in the role of binary prediction. The AdaBoost algorithm focuses on making the optimal forecast directly without modeling the conditional probability of the events. The AdaBoost gets an additive model by iteratively minimizing an exponential loss function. In each iteration, AdaBoost puts more weights on the observations that cannot be predicted correctly using the previous predictors. Moreover, the AdaBoost algorithm is able to solve the classification problem with high-dimensional data which is an advantage to traditional classification methods.

The rest of the chapter is organized as follow. In Section 2.2, we provide a brief introduction of AdaBoost from minimizing the 'exponential loss'. In Section 2.3, we show popular variants of AdaBoost. Section 2.5 gives numerical examples of the boosting algorithms. Section 3.6 compares the mentioned boosting algorithms with Deep Neural Network (DNN) and Logistic Regression with LASSO. Section 2.7 concludes.

2.2 AdaBoost

The algorithm of AdaBoost is as shown in Algorithm 1. Let y be the binary class taking a value in $\{-1, 1\}$ that we wish to predict. Let $f_m(x)$ be the weak learner (weak classifier) for the binary target y that we fit to predict using the high-dimensional covariates x in the mth iteration. Let err_m denote the error rate of the weak learner $f_m(x)$, and $E_w(\cdot)$ denote the weighted expectation (to be defined below) of the variable in the parenthesis with weight w. Note that the error rate $E_w\left[1_{(y \neq f_m(x))}\right]$ is estimated by $err_m = \sum_{i=1}^n w_i 1_{(y_i \neq f_m(x_i))}$ with the weight w_i given by step 2(c) from the previous iteration. n is the number of observations. The symbol $1_{(\cdot)}$ is the indicator function which takes the value 1 if a logical condition inside the parenthesis is satisfied and takes the value 0 otherwise. The symbol $\text{sign}(z) = 1$ if $z > 0$, $\text{sign}(z) = -1$ if $z < 0$, and hence $\text{sign}(z) = 1_{(z>0)} - 1_{(z<0)}$.

Algorithm 1 Discrete AdaBoost (DAB, Freund and Schapire, 1997)

1. Start with weights $w_i = \frac{1}{n}, i = 1, \ldots, n$.
2. For $m = 1$ to M
   (a) For $j = 1$ to k (for each variable)
      i. Fit the classifier $f_{mj}(x_{ij}) \in \{-1, 1\}$ using weights w_i on the training data.
      ii. Compute $err_{mj} = \sum_{i=1}^n w_i 1_{(y_i \neq f_{mj}(x_{ji}))}$.
   (b) Find $\hat{j}_m = \arg\min_j err_{mj}$
   (c) Compute $c_m = \log\left(\frac{1-err_{m,\hat{j}_m}}{err_{m,\hat{j}_m}}\right)$.
   (d) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_{m,\hat{j}_m}(x_{\hat{j}_m,i}))}]$, $i = 1, \ldots, n$, and normalize so that $\sum_{i=1}^n w_i = 1$.
3. Output the binary classifier $\text{sign}[F_M(x)]$ and the class probability prediction $\hat{\pi}(x) = \frac{e^{F_M(x)}}{e^{F_M(x)} + e^{-F_M(x)}}$ where $F_M(x) = \sum_{m=1}^M c_m f_{m,\hat{j}_m}(x_{\hat{j}_m})$.

Remark 1. Note that the presented version of Discrete AdaBoost as well as Real AdaBoost (RAB), LogitBoost (LB) and Gentle AdaBoost (GAB) which will be introduced later in the paper are different from their original version when they are first introduced. The original version of these algorithms only output the class label. In this chapter, we follow the idea of Mease et al. (2007) and modified the algorithms to output both the class label and probability prediction. The probability prediction is attained using

$$\hat{\pi}(x) = \frac{e^{F_M(x)}}{e^{F_M(x)} + e^{-F_M(x)}}, \tag{2.1}$$

where $F_M(x)$ is the sum of the weak learners in the algorithm.

The most widely used weak learner is the classification tree. The simplest classification tree, the stump, takes the following functional form

$$f(x_j, a) = \begin{cases} 1 & x_j > a \\ -1 & x_j < a, \end{cases}$$

where the parameter a is found by minimizing the error rate

$$\min_a \sum_{i=1}^{n} w_i 1\left(y_i \neq f\left(x_{ji}, a\right)\right). \tag{2.2}$$

In addition to the commonly used classification tree weak learners in machine learning literature described above, Discrete AdaBoost, in principle, can take any classifier and boost its performance through the weighted voting scheme. For example, we can also use a one-variable Logistic Regression as a weak learner which we will call the logistic weak learner. Simulation results of Chu et al. (2018a) show that the logistic weak learner generally has better performance than the stump in traditional

econometric models. In the logistic weak learner, we assume the probability

$$\pi(x_j) \equiv P(y=1|x_j) = \frac{e^{x_j\beta}}{1+e^{x_j\beta}}.$$

Let $Y = \frac{y+1}{2} \in \{0,1\}$. We estimate the parameter β by maximizing the weighted logistic log-likelihood function

$$\begin{aligned}
\max_{\beta} \log L &= \log \prod_{i=1}^{n} \left[\left(\frac{e^{x_{ji}\beta}}{1+e^{x_{ji}\beta}} \right)^{Y_i} \left(\frac{1}{1+e^{x_{ji}\beta}} \right)^{1-Y_i} \right]^{w_i} \\
&= \log \prod_{i=1}^{n} \left(\frac{e^{Y_i x_{ji}\beta}}{1+e^{x_{ji}\beta}} \right)^{w_i} \qquad (2.3) \\
&= \sum_{i=1}^{n} \log \left(\frac{e^{Y_i x_{ji}\beta}}{1+e^{x_{ji}\beta}} \right)^{w_i} \\
&= \sum_{i=1}^{n} w_i \left[Y_i x_{ji}\beta - \log\left(1+e^{x_{ji}\beta}\right) \right]. \qquad (2.4)
\end{aligned}$$

Then the resulting logistic weak learner will be

$$f(x_j, \beta, \tau) = \begin{cases} 1 & \pi(x_j, \beta) > 0.5 \\ -1 & \pi(x_j, \beta) < 0.5. \end{cases}$$

Friedman et al. (2000a) show that AdaBoost builds an additive logistic regression model

$$F_M(x) = \sum_{m=1}^{M} c_m f_m(x) \qquad (2.5)$$

via Newton-like updates for minimizing the exponential loss

$$J(F) = E\left(e^{-yF(x)}|x\right). \qquad (2.6)$$

We use greedy method to minimize the exponential loss function iteratively. After

m iterations, the current classifier is denoted as $F_m(x) = \sum_{s=1}^{m} c_s f_s(x)$. In the next iteration, we are seeking an update $c_{m+1} f_{m+1}(x)$ for the function fitted from previous iterations $F_m(x)$. The updated classifier would take the form

$$F_{m+1}(x) = F_m(x) + c_{m+1} f_{m+1}(x).$$

The loss for $F_{m+1}(x)$ will be

$$\begin{aligned} J(F_{m+1}(x)) &= J(F_m(x) + c_{m+1} f_{m+1}(x)) \\ &= E\left[e^{-y(F_m(x) + c_{m+1} f_{m+1}(x))}\right]. \end{aligned} \tag{2.7}$$

Expand w.r.t. $f_{m+1}(x)$

$$\begin{aligned} J(F_{m+1}(x)) &\approx E\left[e^{-yF_m(x)}\left[1 - yc_{m+1} f_{m+1}(x) + \frac{y^2 c_{m+1}^2 f_{m+1}^2(x)}{2}\right]\right] \\ &= E\left[e^{-yF_m(x)}\left(1 - yc_{m+1} f_{m+1}(x) + \frac{c_{m+1}^2}{2}\right)\right]. \end{aligned}$$

The last equality holds since $y \in \{-1, 1\}$, $f_{m+1}(x) \in \{-1, 1\}$, and $y^2 = f_{m+1}^2(x) = 1$. $f_{m+1}(x)$ only appears in the second term in the parenthesis, so minimizing the loss function (2.7) w.r.t. $f_{m+1}(x)$ is equivalent to maximizing the second term in the parenthesis which results in the following conditional expectation

$$\max_f E\left[e^{-yF_m(x)} yc_{m+1} f_{m+1}(x) \,|x\right].$$

For any $c > 0$ (we will prove this later), we can omit c_{m+1} in the above objective function

$$\max_f E\left[e^{-yF_m(x)} y f_{m+1}(x) \,|x\right].$$

To compare it with the Discrete AdaBoost algorithm, here we define weight $w = w(y,x) = e^{-yF_m(x)}$. Later we will see that this weight w is equivalent to that shown in the Discrete AdaBoost algorithm. So the above optimization can be seen as maximizing a weighted conditional expectation

$$\max_f E_w \left[y f_{m+1}(x) \,|x\right] \tag{2.8}$$

where $E_w(y|x) := \frac{E(wy|x)}{E(w|x)}$ refers to a weighted conditional expectation. Note that (2.8)

$$\begin{aligned} & E_w \left[y f_{m+1}(x) \,|x\right] \\ = \; & P_w(y=1|x) f_{m+1}(x) - P_w(y=-1|x) f_{m+1}(x) \\ = \; & \left[P_w(y=1|x) - P_w(y=-1|x)\right] f_{m+1}(x) \\ = \; & E_w(y|x) f_{m+1}(x). \end{aligned}$$

where $P_w(y|x) = \frac{E(w|y,x)P(y|x)}{E(w|x)}$. Solve the maximization problem (2.8). Since $f_{m+1}(x)$ only takes 1 or -1, it should be positive whenever $E_w(y|x)$ is positive and -1 whenever $E_w(y|x)$ is negative. The solution for $f_{m+1}(x)$ is

$$f_{m+1}(x) = \begin{cases} 1 & E_w(y|x) > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Next, minimize the loss function (2.7) w.r.t. c_{m+1}

$$\begin{aligned} c_{m+1} &= \arg\min_{c_{m+1}} E_w\left(e^{-c_{m+1}yf_{m+1}(x)}\right) \\ E_w\left(e^{-c_{m+1}yf_{m+1}(x)}\right) &= P_w\left(y = f_{m+1}(x)\right)e^{-c_{m+1}} + P_w\left(y \neq f_{m+1}(x)\right)e^{c_{m+1}} \\ \frac{\partial E_w\left(e^{-cyf_{m+1}(x)}\right)}{\partial c} &= -P_w\left(y = f_{m+1}(x)\right)c_{m+1}e^{-c_{m+1}} + P_w\left(y \neq f_{m+1}(x)\right)c_{m+1}e^{c_{m+1}} \end{aligned}$$

Let

$$\frac{\partial E_w\left(e^{-c_{m+1}yf_{m+1}(x)}\right)}{\partial c_{m+1}} = 0,$$

and we have

$$P_w\left(y = f_{m+1}(x)\right)c_{m+1}e^{-c_{m+1}} = P_w\left(y \neq f_{m+1}(x)\right)c_{m+1}e^{c_{m+1}},$$

Solve for c_{m+1}, we obtain

$$c_{m+1} = \frac{1}{2}\log\frac{P_w\left(y = f_{m+1}(x)\right)}{P_w\left(y \neq f_{m+1}(x)\right)} = \frac{1}{2}\log\left(\frac{1 - err_{m+1}}{err_{m+1}}\right),$$

where $err_{m+1} = P_w\left(y \neq f_{m+1}(x)\right)$ is the error rate of $f_{m+1}(x)$. Note that $c_{m+1} > 0$ as long as the error rate is smaller than 50%. Our assumption $c_{m+1} > 0$ holds for any learner that is better than random guessing.

Now we have finished the steps of one iteration and can get our updated classifier by

$$F_{m+1}(x) \leftarrow F_m(x) + \left(\frac{1}{2}\log\left(\frac{1 - err_{m+1}}{err_{m+1}}\right)\right)f_{m+1}(x).$$

Note that in the next iteration, the weight we defined w_{m+1} will be

$$w_{m+1} = e^{-yF_{m+1}(x)} = e^{-y(F_m(x)+c_{m+1}f_{m+1}(x))} = w_m \times e^{-c_{m+1}f_{m+1}(x)y}.$$

Since $-yf_{m+1}(x) = 2 \times 1_{\{y \neq f_{m+1}(x)\}} - 1$, the update is equivalent to

$$w_{m+1} = w_m \times e^{\left(\log\left(\frac{1-err_{m+1}}{err_{m+1}}\right) 1_{[y \neq f_{m+1}(x)]}\right)} = w_m \times \left(\frac{1-err_{m+1}}{err_{m+1}}\right)^{1_{[y \neq f_{m+1}(x)]}}.$$

Thus the function and weights update are of an identical form to those used in AdaBoost. AdaBoost could do better than any single weak classifier since it iteratively minimizes the loss function via a Newton-like procedure. Interestingly, the function $F(x)$ from minimizing the exponential loss is the same as maximizing a logistic log-likelihood. Let

$$\begin{aligned} J(F(x)) &= E\left[E\left(e^{-yF(x)}|x\right)\right] \\ &= E\left[P(y=1|x)\, e^{-F(x)} + P(y=-1|x)\, e^{F(x)}\right]. \end{aligned}$$

Taking derivative w.r.t. $F(x)$ and making it equal to zero, we obtain

$$\begin{aligned} \frac{\partial E\left(e^{-yF(x)}|x\right)}{\partial F(x)} &= -P(y=1|x)\, e^{-F(x)} + P(y=-1|x)\, e^{F(x)} = 0 \\ F^*(x) &= \frac{1}{2}\log\left[\frac{P(y=1|x)}{P(y=-1|x)}\right]. \end{aligned}$$

Moreover, if the true probability

$$P(y=1|x) = \frac{e^{2F(x)}}{1+e^{2F(x)}},$$

for $Y = \frac{y+1}{2}$, the log-likelihood is

$$E(\log L|x) = E\left[2YF(x) - \log\left(1+e^{2F(x)}\right)|x\right].$$

The solution $F^*(x)$ that maximize the log-likelihood must equals the $F(x)$ in the

true model $P(y=1|x) = \frac{e^{2F(x)}}{1+e^{2F(x)}}$. Hence,

$$
\begin{aligned}
e^{2F^*(x)} &= P(y=1|x)\left(1+e^{2F^*(x)}\right) \\
e^{2F^*(x)} &= \frac{P(y=1|x)}{1-P(y=1|x)} \\
F^*(x) &= \frac{1}{2}\log\left[\frac{P(y=1|x)}{P(y=-1|x)}\right].
\end{aligned} \tag{2.9}
$$

AdaBoost that minimizes the exponential loss yield the same solution as Logistic Regression that maximizes the logistic log-likelihood.

2.3 Extensions to AdaBoost Algorithms

In this section, we introduce extensions of Discrete AdaBoost, namely Real AdaBoost (RAB), LogitBoost (LB) and Gentle AdaBoost (GAB), and discuss how some aspects of the DAB may be modified to yield RAB, LB, and GAB. In the last section, we learned that Discrete AdaBoost minimizes an exponential loss via iteratively adding a binary weaker learner to the pool of weak learners. The addition of a new weak learner can be seen as taking a step on the direction that loss function descents in the Newton method. There are two major ways to extend the idea of Discrete AdaBoost. One focuses on making the minimization method more efficient by adding a more flexible weak learner. The other is to use different loss functions that may lead to better results. Next, we give an introduction to several extensions of Discrete AdaBoost.

2.3.1 Real AdaBoost

Algorithm 2 Real AdaBoost (RAB, Friedman et al. 2000)

1. Start with weights $w_i = \frac{1}{n}, i = 1, \ldots, n$.
2. For $m = 1$ to M
   (a) For $j = 1$ to k (for each variable)
      i. Fit the classifier to obtain a class probability estimate $p_m(x_j) = \hat{P}_w(y = 1|x_j) \in [0, 1]$ using weights w_i on the training data.
      ii. Let $f_{mj}(x_j) = \frac{1}{2} \log \frac{p_m(x_j)}{1-p_m(x_j)}$.
      iii. Compute $err_{mj} = \sum_{i=1}^{n} w_i 1_{(y_i \neq \texttt{sign}(f_{mj}(x_{ji})))}$.
   (b) Find $\hat{j}_m = \arg\min_j err_{mj}$.
   (c) Set $w_i \leftarrow w_i \exp\left[-y_i f_{m,\hat{j}_m}(x_{\hat{j}_m,i})\right]$, $i = 1, \ldots, n$, and normalize so that $\sum_{i=1}^{n} w_i = 1$.
3. Output the classifier $\text{sign}[F_M(x)]$ and the class probability prediction $\hat{\pi}(x) = \frac{e^{F_M(x)}}{e^{F_M(x)}+e^{-F_M(x)}}$ where $F_M(x) = \sum_{m=1}^{M} f_m(x)$.

Real AdaBoost focuses solely on improving the minimization procedure of Discrete AdaBoost. In Real AdaBoost, the weak learners are continuous comparing to Discrete AdaBoost where the weak learners are binary (discrete). Real AdaBoost is minimizing the exponential loss with continuous updates where Discrete AdaBoost minimizes the exponential loss with discrete updates. Hence, Real AdaBoost is more flexible with the step size and direction of the minimization and minimizes the exponential loss faster and more accurately. However, Real AdaBoost also imposes the restriction that the classifier must produce a probability prediction which reduces the flexibility of the model. As we shall see in the numerical examples, Real AdaBoost may achieve a larger in-sample training error due to the flexibility of its model. On the other hand, this also reduces the chances of fitting and would, in the end, achieve a smaller out-of-sample test error.

2.3.2 LogitBoost

Friedman et al. (2000a) propose LogitBoost by minimizing the Bernoulli log-likelihood via an adaptive Newton algorithm for fitting an additive logistic model. LogitBoost extends Discrete AdaBoost in two ways. First, it uses the Bernoulli log-likelihood instead of the exponential function as the loss function. Furthermore, it updates the classifier by adding a linear model instead of a binary weak learner.

Algorithm 3 LogitBoost (LB, Friedman et al., 2000a)

1. Start with weights $w_i = \frac{1}{n}, i = 1, \ldots, n$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. For $m = 1$ to M
   (a) Compute the working response and weights
   $$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))} \tag{2.10}$$
   $$w_i = p(x_i)(1 - p(x_i)) \tag{2.11}$$
   (b) For $j = 1$ to k (for each variable)
      i. Fit the function $f_{mj}(x_{ji})$ by a weighted least-squares regression of z_i to $x_j i$ using weights w_i on the training data.
      ii. Compute $err_{mj} = 1 - R^2$ from the weighted least-squares regression.
   (c) Find $\hat{j}_m = \arg\min_j err_{mj}$
   (d) Update $F(x) \leftarrow F(x) + \frac{1}{2} f_{m,\hat{j}}(x_{\hat{j}})$ and $p(x) \leftarrow \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}}$, $i = 1, \ldots, n$.
3. Output the classifier $\text{sign}[F_M(x)]$ and the class probability prediction $\hat{\pi}(x) = \frac{e^{F_M(x)}}{e^{F_M(x)} + e^{-F_M(x)}}$ where $F_M(x) = \sum_{m=1}^{M} f_{m,\hat{j}_m}(x_{\hat{j}_m})$.

In LogitBoost, continuous weak learners are used similar to Real AdaBoost. However, LogitBoost specified the use of linear weak learner while Real AdaBoost allows any weak learner that returns a probability between zero and one. A bigger and more fundamental difference here is that LogitBoost uses the Bernoulli log-likelihood

as loss function instead of the exponential loss. Hence, LogitBoost is more similar to Logistic Regression than Discrete AdaBoost and Real AdaBoost. As we will see in the simulation result, LogitBoost has the smallest in-sample training error but the largest out-of-sample test error. This implies that while LogitBoost is the most flexible of the four, it suffers the most from over-fitting.

2.3.3 Gentle AdaBoost

Algorithm 4 Gentle AdaBoost (GAB, Friedman et al., 2000a)

1. Start with weights $w_i = \frac{1}{n}, i = 1, \ldots, n$.
2. For $m = 1$ to M
   (a) For $j = 1$ to k (for each variable)
      i. Fit the regression function $f_{mj}(x_{ij})$ by weighted least-squares of y_i on x_i using weights w_i on the training data.
      ii. Compute $err_{mj} = 1 - R^2$ from the weighted least-squares regression.
   (b) Find $\hat{j}_m = \arg\min_j err_{mj}$
   (c) Set $w_i \leftarrow w_i \exp[-y_i f_{m,\hat{j}_m}(x_{\hat{j}_m,i})]$, $i = 1, \ldots, n$, and normalize so that $\sum_{i=1}^{n} w_i = 1$.
3. Output the classifier $\text{sign}[F_M(x)]$ and the class probability prediction $\hat{\pi}(x) = \frac{e^{F_M(x)}}{e^{F_M(x)}+e^{-F_M(x)}}$ where $F_M(x) = \sum_{m=1}^{M} f_{m,\hat{j}_m}(x_{\hat{j}_m})$.

Gentle AdaBoost extends Discrete AdaBoost in the sense that it allows each weak learner to be a linear model. This is similar to LogitBoost and more flexible than Discrete AdaBoost and Real AdaBoost. However, it is closer to Discrete AdaBoost and Real AdaBoost than LogitBoost in the sense that Gentle AdaBoost, Discrete AdaBoost, and Real AdaBoost all minimize the exponential loss while LogitBoost minimizes the Bernoulli log-likelihood. Another point that Gentle AdaBoost is more similar to Real AdaBoost than Discrete AdaBoost is that since the weak learners are

continuous, there is no need to find an optimal step size for each iteration because the weak learner is already optimal. As we will see in the simulation results, Gentle Boost often lies between Real AdaBoost and LogitBoost in terms of in-sample training error and out-of-sample test error.

Note that manually transforming the sum of all weak learners $F(x)$ into probability prediction using equation 2.9 would lead to the same result as directly output the probability prediction from the package as in the second line.

2.4 Alternative Classification Methods

Apart from Boosting algorithms, we also consider Deep Neural Network, Logistic Regression and semiparametric single-index model as alternative methods to obtain a predictor of y. Deep Neural Network is able to deal with high-dimensional data. For Logistic Regression, we have to select useful information from noises. Hence, a shrinkage parameter is used with the logistic log-likelihood which we call LASSO. Semiparametric single-index model is an extension to the parametric single-index model such as Logistic Regression. It relaxes the parametric assumptions and uses the kernel function to fit the data locally. For high-dimensional problem, we use SIM-RODEO to select useful explanation variables for semiparametric single-index models.

2.4.1 Deep Neural Network

Deep Neural Network is undoubtedly one of the most state-of-the-art classification methods. The model is similar to a multi-stage regression or classification model. The idea is to build a flexible nonlinear statistical model consisted of several layers and

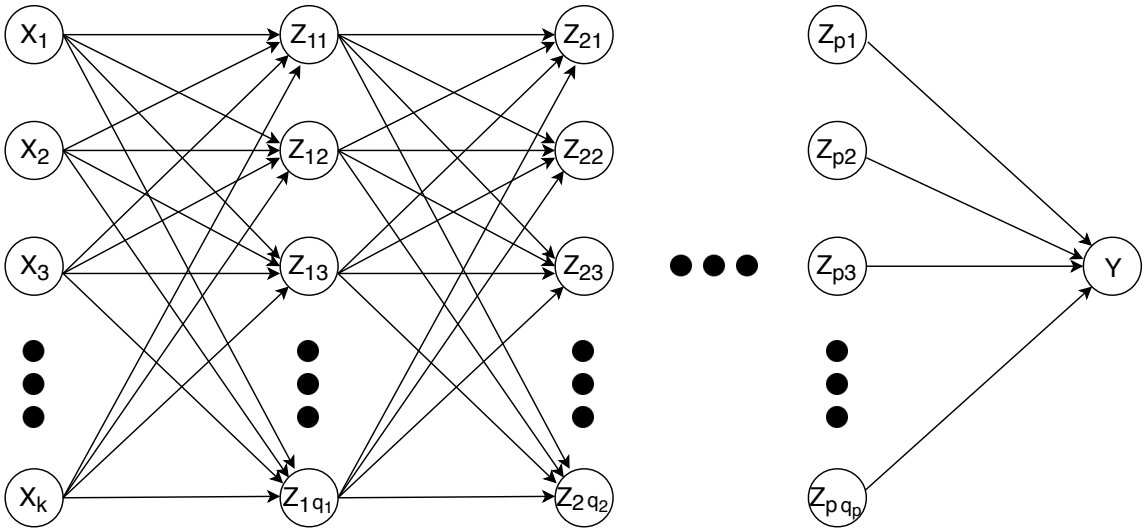


Figure 2.1: Diagram of Deep Neural Network

each layer consists of neurons as in Figure 2.1.

For binary classification, there is only one output Y that is the class probability or class label. Since the transformation from class probability to class label is straightforward, we focus on the case where the output is the class probability. The layer labeled X is the input layer which contains all the explanatory variables in the data set. Note that the number of explanatory variables k is allowed to be extremely large (larger than the number of observations) as in high-dimensional settings. The layers labeled Z are the hidden layers. The number of hidden layers p can be arbitrarily set by the user and each hidden layer can contain arbitrarily many neurons denoted by q_t where t stands for the tth hidden layer.

The output z_{ts} of the sth neuron in the tth hidden layer is normally a single-index function $g(\alpha_{ts} + \beta'_{ts} Z_{t-1})$ where α is a scalar, β is a vector of same length q_{t-1} as the number of neurons in the $(t-1)$th hidden layer or the input layer if $t = 1$ and $Z_{t-1} = (z_{t-1,1}, z_{t-1,2}, \ldots, z_{t-1,q_{t-1}})$ is a vector of outputs from all neurons of the $(t-1)$th hidden layer or the input layer if $t = 1$. Similarly, the output layer of the model is also chosen to be a single-index function of the outputs of the last hidden

layer. Hence,

$$z_{1s} = g(w_{01s} + w'_{1s}X) \tag{2.12}$$

$$z_{ts} = g(w_{0ts} + w'_{ts}Z_{t-1}) \tag{2.13}$$

$$Y = \hat{\pi}(x) = f(w_0 + w'Z_t). \tag{2.14}$$

The function $g(v)$ is called the activation function. It is often chosen to be a sigmoid. Popular choices are the Rectified Linear Unit (ReLU)

$$g(v) = \max(0, v)$$

and the logistic function

$$g(v) = \frac{1}{1 + e^{-v}}.$$

The function $f(v)$ in the output layer can also be a sigmoid. In addition to the ReLU and logistic function, the identity function can also be used as the output function.

Since the activation function, output function, and the number of hidden layers and neurons are all chosen by the user prior to fitting the model, the only parameters to be determined by the data are the weights α's and β's. We choose the best values for α's and β's to minimize a given loss function. For binary classification, the squared error loss

$$L(w) = \sum_i (y_i - \hat{\pi}(x_i))^2$$

and the cross-entropy

$$L(w) = -\sum_i y_i \log \hat{\pi}(x_i)$$

are often used. The minimization procedure of Deep Neural Network is often time-consuming. Moreover, convergence and optimality cannot be guaranteed. Hence,

multiple attempts need to be made for a single problem. Two techniques, the stochastic gradient descent and the back-propagation, are often used for minimization of Deep Neural Network. Fortunately, we do not have to worry about the implementation of the minimization procedure since packages are available in `R`.

Remark 2. Note that the class probability can be converted to class label easily by the rule $\hat{Y} = 1\left(\pi(\hat{x}) > 0.5\right)$ where $1(\cdot)$ is the indicator function.

2.4.2 Logistic Regression with LASSO

In traditional econometrics, the most used classification and probability prediction method should be the Logistic Regression. The Logistic Regression assumes that the probability that the output variable $Y = \frac{y+1}{2} \in \{0, 1\}$ takes value 1 follows a logistic function of x. That is

$$\pi(x) \equiv P(Y = 1|x) = \frac{1}{1 + e^{-x\beta}}.$$

Given a sample data of y and x, the likelihood of the sample can be rewritten as

$$L(\beta) = \prod_i \left(\frac{1}{1 + e^{-x_i\beta}}\right)^{Y_i} \left(\frac{1}{1 + e^{x_i\beta}}\right)^{1-Y_i}. \tag{2.15}$$

Taking the log transformation, the log-likelihood is

$$\begin{aligned}
\log L(\beta) &= \log\left(\prod_i \left(\frac{1}{1+e^{-x_i\beta}}\right)^{Y_i}\left(\frac{1}{1+e^{x_i\beta}}\right)^{1-Y_i}\right) && (2.16)\\
&= \sum_i \log\left(\left(\frac{1}{1+e^{-x_i\beta}}\right)^{Y_i}\left(\frac{1}{1+e^{x_i\beta}}\right)^{1-Y_i}\right) && (2.17)\\
&= \sum_i \log\left(\frac{1}{1+e^{-x_i\beta}}\right)^{Y_i} + \log\left(\frac{1}{1+e^{x_i\beta}}\right)^{1-Y_i} && (2.18)\\
&= \sum_i Y_i x_i\beta - \log\left(1+e^{-x_i\beta}\right). && (2.19)
\end{aligned}$$

Because of the high-dimensional feature of our problem, we have to control the number of explanatory variables included in the model. Hence, an L_1 penalty a.k.a LASSO penalty is added to the log-likelihood as a penalty to including more explanatory variables in the model. Logistic Regression with LASSO minimizes the negative logistic log-likelihood (2.4) with a Lasso penalty as below

$$\min -\sum_{t=1}^{N}\left[Y_i x_i\beta - \log\left(1+e^{x_i\beta}\right)\right] + \lambda\left|\beta\right|_1. \qquad (2.20)$$

A well-known package called *glmnet* package provided by Hastie and Qian uses a quadratic approximation to the log-likelihood, and then coordinate descent on the resulting penalized weighted least-squares problem. And it is so far the most trustworthy package in `R` for Logistic Regression with LASSO. For binary classification, we use the estimated β to construct a logistic probability model for y. Then, get our prediction from the model. If $\hat{\pi}(x) > 0.5$, the predicted class will be 1. And if $\hat{\pi}(x) < 0.5$, the predicted class will be 0.

2.4.3 Semiparametric Single-Index Model

Chu et al. (2018b) consider a standard single index model,

$$y = m\left(x'\beta\right) + u, \tag{2.21}$$

where $\beta = (\beta_1, \ldots, \beta_k)$ is a vector of coefficients. Under the sparsity condition, we assume that $\beta_j \neq 0$ for $j \leq r$ and $\beta_j = 0$ for $j > r$. We also assume that the random errors u are independent. However, we allow the presence of heteroskedasticity to encompass a large category of models for binary prediction, e.g. Logit and Probit models. The kernel estimator (Ichimura, 1993) we use is as shown below

$$\hat{m}\left(x'\beta; h\right) = \frac{\sum_{i=1}^{n} y_i K\left(\frac{X_i'\beta - x'\beta}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i'\beta - x'\beta}{h}\right)}, \tag{2.22}$$

where $K\left(\cdot\right)$ is a kernel function. The semiparametric kernel regression looks for the best β and h to minimize a weighted squared error loss. However, exact identification is not available. If one blows up β and θ simultaneously by multiplying the same constant, the kernel estimator would yield identical estimates and losses. The standard identification approach is to set the first element of β to be 1 (Ichimura, 1993).

In terms of variable selection and prediction, we only need to focus on finding the best $\theta \equiv \frac{\beta}{h}$. Hence, we can simplify the estimator to

$$\hat{m}\left(x'\theta\right) = \frac{\sum_{i=1}^{n} y_i K\left(X_i'\theta - x'\theta\right)}{\sum_{i=1}^{n} K\left(X_i'\theta - x'\theta\right)}. \tag{2.23}$$

The basic idea of the SIM-RODEO is to view the local bandwidth selection as a variable selection in sparse semiparametric single index model. The SIM-RODEO algorithm amplifies the inverse of the bandwidths for relevant variables while keeping

the inverse of the bandwidths of irrelevant variables relatively small. The SIM-RODEO algorithm is greedy as it solves for the locally optimal path choice at each iteration. It can also be shown to attain the consistency in mean square error when it is applied for sparse semiparametric single index models. SIM-RODEO is able to distinguish truly relevant explanatory variables from noisy irrelevant variables and gives a consistent estimator of the regression function. In addition, the algorithm is fast to finish the greedy steps.

Now we derive the RODEO for Single Index Models. First we introduce some notation. Let

$$W_x = \begin{pmatrix} K\left(X_1'\theta - x'\theta\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K\left(X_n'\theta - x'\theta\right) \end{pmatrix} \tag{2.24}$$

where $K(\cdot)$ is the Gaussian kernel. The standard Ichimura (1993) estimator takes the form

$$\hat{m}\left(x'\theta\right) = \frac{\sum_{i=1}^{n} y_i K\left(X_i'\theta - x'\theta\right)}{\sum_{i=1}^{n} K\left(X_i'\theta - x'\theta\right)} = \left(\iota' W_x \iota\right)^{-1} \iota' W_x y. \tag{2.25}$$

The derivative of the estimator Z_j with respect to θ_j is

$$\begin{aligned} Z_j &\equiv \frac{\partial \hat{m}\left(x'\theta\right)}{\partial \theta_j} && (2.26) \\ &= \left(\iota' W_x \iota\right)^{-1} \iota' \frac{\partial W_x}{\partial \theta_j} y - \left(\iota' W_x \iota\right)^{-1} \iota' \frac{\partial W_x}{\partial \theta_j} \iota \left(\iota' W_x \iota\right)^{-1} \iota' W_x y \\ &= \left(\iota' W_x \iota\right)^{-1} \iota' \frac{\partial W_x}{\partial \theta_j} \left(y - \iota \hat{m}\left(x'\theta\right)\right). && (2.27) \end{aligned}$$

For the ease of computation, let

$$L_j = \begin{pmatrix} \frac{\partial \log K\left(X_1'\theta - x'\theta\right)}{\partial \theta_j} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial \log K(X_n'\theta - x'\theta)}{\partial \theta_j} \end{pmatrix}. \tag{2.28}$$

Note that

$$\frac{\partial W_x}{\partial \theta_j} = W_x L_j, \tag{2.29}$$

which appears in equation (4.16). With the Gaussian kernel, $K(t) = e^{-\frac{t^2}{2}}$, then L_j becomes

$$\begin{aligned} L_j &= \begin{pmatrix} -\frac{1}{2}\frac{\partial\left(X_1'\theta - x'\theta\right)^2}{\partial \theta_j} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -\frac{1}{2}\frac{\partial (X_n'\theta - x'\theta)^2}{\partial \theta_j} \end{pmatrix} \\ &= \begin{pmatrix} -\left(X_1'\theta - x'\theta\right)\left(X_{1j} - x_j\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -\left(X_n'\theta - x'\theta\right)\left(X_{nj} - x_j\right) \end{pmatrix}, \end{aligned}$$

where X_{1j} and X_{nj} are the jth elements of vectors X_1 and X_n. And x_j is the jth element of vector x. To simplify the notation, let $B_x = \left(\iota' W_x \iota\right)^{-1} \iota' W_x$. Then, the derivative Z_j becomes

$$\begin{aligned} Z_j &= \left(\iota' W_x \iota\right)^{-1} \iota' \frac{\partial W_x}{\partial \theta_j}\left(y - \iota \hat{m}\left(x'\theta\right)\right) \\ &= B_x L_j \left(I - \iota B_x\right) y \\ &\equiv G_j\left(x, \theta\right) y. \end{aligned} \tag{2.30}$$

Next, we give the conditional expectation and variance of Z_j.

$$
\begin{aligned}
Z_j &= G_j(x,\theta)\, y = G_j(x,\theta)\,(m(x'\beta)+u), && (2.31)\\
E(Z_j|X) &= E(G_j(x,\theta)\,(m(x'\beta)+u)\,|X) = G_j(x,\theta)\, m(x'\beta), && (2.32)\\
\mathrm{Var}(Z_j|X) &= \mathrm{Var}(G_j(x,\theta)\,(m(x'\beta)+u)\,|X) = \boldsymbol{\sigma}' G_j(x,\theta)' G_j(x,\theta)\,\boldsymbol{\sigma}, && (2.33)
\end{aligned}
$$

where $\boldsymbol{\sigma} = (\sigma(u_1), \ldots, \sigma(u_n))'$ is the vector of standard deviations of u. In the algorithm, it is necessary to insert an estimate of σ. Since we allow the errors to be heteroskedastic as in Logit and Probit models and estimate $\sigma(u_i)$ using the estimator $\hat{\sigma}(u_i) = m(x_i'\hat{\theta})(1 - m(x_i'\hat{\theta}))$.

SIM-RODEO is described in Algorithm 5, which is a modified algorithm of RODEO (Lafferty and Wasserman, 2008b).

Algorithm 5 SIM-RODEO (Chu et al., 2018b)

1. Select a constant $0 < \alpha < 1$ and the initial value

$$\theta_0 = c_0 \log\log n$$

   where c_0 is sufficiently small. Compute Z_j with $\theta_j = \theta_0$ for all j.

2. Initialize the coefficients θ, and activate all covariates:

   (a) $\theta_j = \begin{cases} \theta_0 & Z_j > 0 \\ -\theta_0 & \text{otherwise,} \end{cases} \quad j = 1, \ldots, k.$

   (b) $\mathcal{A} = \{1, \ldots, k\}$.

3. While $\mathcal{A} \neq \emptyset$ is nonempty, do for each $j \in \mathcal{A}$:

   (a) Compute Z_j and $s_j = \sqrt{\text{Var}\left(Z_j | X\right)}$ using (4.19) and (4.22) respectively.

   (b) Compute the threshold $\lambda_j = s_j \sqrt{2 \log n}$.

   (c) If $|Z_j| > \lambda_j$, then set $\theta_j \leftarrow \frac{\theta_j}{\alpha}$; Otherwise, remove j from $\mathcal{A}$ (i.e., $\mathcal{A} \leftarrow \mathcal{A} - \{j\}$).

4. Obtain $\hat{\theta} = (\theta_1, \ldots, \theta_k)$. Output the class probability prediction $\hat{\pi}(x) = \hat{m}\left(x'\hat{\theta}\right)$ and the classifier $F(x) = 1_{(\hat{\pi}(x) > 0.5)}$.

We start by setting $\theta_j = \theta_0$ that is close to zero. Hence, $(X_i'\theta - x'\theta)$ are close to zero and $K\left(X_i'\theta - x'\theta\right)$ are close to $K\left(0\right)$. This means our estimator starts with the simple average of all observations, $\bar{y}$. If the derivative of θ_j is statistically different from zero. We amplify θ_j. If x_j is indeed a relevant explanatory variable, then the weights $K\left(X_i'\theta - x'\theta\right)$ change according to x_j. The estimator will give higher weights to observations close to $x'\theta$ and lower weights to observations away from $x'\theta$.

2.5 Monte Carlo

In this section, we demonstrate the above DAB, RAD, LB, and GB via Monte Carlo simulation designs. We construct the two DGPs to check the finite sample

properties of the Boosting algorithms. DGP1 is a binary logistic model where y follows a Bernoulli distribution with probability

$$\pi(x) \equiv \frac{1}{1+e^{-x\beta}}$$

to be 1 and $1-\pi(x)$ to be -1 where

$$\underset{n\times k}{x} \sim N\left(0, \frac{\Sigma}{\beta'\Sigma\beta}\right),\ \Sigma_{ij} = \rho^{|i-j|},$$

$$n = 100, k = \{2, 20\} \text{ and } \rho \in \{0\}.$$

We have two settings for the β. In the low-dimension case ($k = 2$) , we let

$$\beta = (1, 1).$$

In the high-dimension case ($k = 20$), we let $\beta = (\beta_1, \ldots, \beta_k)$ where

$$\beta_i = 0.9^i. \tag{2.34}$$

that decrease exponentially. Hence, most of the β's are very close 0.

DGP2 is the circle model. Here we have two settings for the circle model. In the low dimension case, only the two relevant x's are used to train the models as shown in the toy demo in previous sections. In the high dimension (sparse) case, three irrelevant x's are added in addition to the two relevant ones. Table 2.1 and Table 2.2 show the in-sample training error and the out-of-sample test error in the two cases for different methods.

To construct the training and testing samples, we randomly generate x using the above distribution and calculate $\pi(x)$. To generate the random variable y based on

x, we first generate a random variable ϵ that follows uniform distribution between $[0, 1]$. Next, we compare ϵ with $\pi(x)$. There is a probability of $\pi(x)$ that ϵ is smaller than $\pi(x)$ and a probability $1 - \pi(x)$ otherwise. Hence, we set

$$y = \begin{cases} 1 & \epsilon < \pi(x) \\ -1 & \epsilon > \pi(x). \end{cases}$$

Given a set of observations $\{(x, y)\}$, we compare the average loss (classification error) achieved by using different methods. The formula for the average loss is as below.

$$ErrorRate = \frac{1}{n} \sum 1\left(y_i \neq \texttt{sign}\left(F_M(x_i)\right)\right), \tag{2.35}$$

where n is the number of observations in the set.

To evaluate the algorithms, first we train our predictors with the training data of size $n = 100$. Then, we use a testing data set that contains 100 new observations of (x, y) to compute the average loss (2.35) achieved by the Boosting algorithms, Deep Neural Network, Logistic Regression and semiparametric Single-Index Model for out-of-sample evaluations. The boosting algorithms are component-wise versions of the four methods as shown before. The alternative methods we have, Deep Neural Network, Logistic Regression with LASSO penalty and semiparametric single-index model with SIM-RODEO considers all variables at the same time. The number of Monte Carlo repetition for each DGP is 1000.

The results are shown below.

From the simulation results, we can see that the four boosting methods work well in both the circle model and the logistic model. LogitBoost has the smallest training

Table 2.1: Error Rate of Low Dimension Circle Model

	Train Error	Test Error
Discrete AdaBoost	0.0820	0.2053
Real AdaBoost	0.0853	0.2038
LogitBoost	0.0602	0.2090
Gentle AdaBoost	0.0718	0.2062
Deep Neural Network	0.2601	0.3533
Logistic Regression	0.3586	0.3573
SIM-RODEO	0.2986	0.3421

Table 2.2: Error Rate of High Dimension (Sparse) Circle Model

	Train Error	Test Error
Discrete AdaBoost	0.0202	0.2203
Real AdaBoost	0.0295	0.2165
LogitBoost	0.0081	0.2232
Gentle AdaBoost	0.0133	0.2208
Deep Neural Network	0.2838	0.4017
Logistic Regression	0.3569	0.3572
SIM-RODEO	0.3542	0.3541

Table 2.3: Error Rate of Low Dimension Logistic Model

	Train Error	Test Error
Discrete AdaBoost	0.1431	0.3129
Real AdaBoost	0.1519	0.3120
LogitBoost	0.1302	0.3160
Gentle AdaBoost	0.1339	0.3154
Deep Neural Network	0.2304	0.3090
Logistic Regression	0.2773	0.3083
SIM-RODEO	0.3069	0.3415

Table 2.4: Error Rate of High Dimension (Sparse) Logistic Model

	Train Error	Test Error
Discrete AdaBoost	0.0007	0.3217
Real AdaBoost	0.0015	0.3215
LogitBoost	0.00007	0.3214
Gentle AdaBoost	0.0001	0.3204
Deep Neural Network	0.0523	0.3172
Logistic Regression	0.2328	0.3432
SIM-RODEO	0.3580	0.3971

error among all four boosting algorithms as well as the largest testing error. On the other hand, Real AdaBoost has the largest training error as well as the smallest testing error. Similar rules apply to the other two boosting methods. Smaller training errors implies larger testing errors. This is evidence of overfitting which is related to the hyper-parameters in the boosting algorithms. If the number of boosting iterations is small, then we will have a larger training error but less risk of overfitting. On the other hand, if we have more boosting iterations, then the boosting methods will fit the training data better but raise higher risk on overfitting. The number of iterations in the boosting algorithms are fixed by the users. However, cross-validation could be used to determine the optimal number of iterations.

As for the alternative methods, Deep Neural Network works better in the logistic model than the circle model. This is a result of the set-up of the Deep Neural Network. We use the logistic function as the activation function and output function, and the entropy as the loss function. The set-up will give better results when the logistic model is the true model. For the circle model, Deep Neural Network gives a comparable result to the Logistic Regression in the low-dimension case. However, the result is much worse for the high-dimension case. Again, this could be a result of our set-up of the Deep Neural Network. We acknowledge that the Deep Neural Network is highly flexible with lots of hyper-parameters Different set-up of the model may lead

to dramatically distinct results. Our setting by no means is the optimal one and Deep Neural Network could perform better with a different set-up.

For Logistic Regression, it works best in the low-dimension logistic model as all parametric assumptions are satisfied. However, in the high-dimension case, Logistic Regression will have a larger bias due to the need to shrink the coefficients of irrelevant variables to zero. To fix this bias, one may try the De-biased Machine Learning method (Chernozhukov et al., 2018).

2.6 Applications

In the application, we use the FRED monthly data `https://research.stlouisfed.org/econ/mccracken/fred-databases/` to predict the moving direction of real personal income in the United States. After removing the observations with missing values, we obtain 341 effective observations with a sample period starting from September, 1989 to January 2018. We use 125 variables which are all variables in the data except for the Consumer Sentiment Index that is only available quarterly and New Orders for Consumer Goods which has too many missing data. We generate the direction of the real personal income as our dependent variable and take the lag of the dependent variable as one explanatory variable. Hence, we have in total $k = 126$ explanatory variables and $(341 - 1 = 340)$ observations. We use rolling training samples with window width $W = 100$ and predict the one month ahead moving direction. We have $(n = 340 - W = 240)$ subsamples and predictions.

The results are very similar to the simulation results for logistic models. The boosting methods have very small in-sample training errors. However, the out-of-sample testing error is much larger than the alternatives. This may indicate that the

Table 2.5: Error Rate of Application

	Train Error	Test Error
Discrete AdaBoost	0.0028	0.3125
Real AdaBoost	0.0407	0.3291
LogitBoost	0.0003	0.3041
Gentle AdaBoost	0.0020	0.3083
Deep Neural Network	0.2389	0.2666
Logistic Regression	0.2479	0.2708
SIM-RODEO	0.2257	0.2958

boosting algorithms are overfitting the model.

2.7 Conclusions

This chapter shows recently developed methods for high-dimensional binary classification and probability prediction. We start by introducing four component-wise boosting methods, namely component-wise Discrete AdaBoost, component-wise Real AdaBoost, component-wise LogitBoost and component-wise Gentle AdaBoost. Discrete AdaBoost, Real AdaBoost, and Gentle AdaBoost minimize the exponential loss via Newton-like procedures. LogitBoost minimizes the Bernoulli log-likelihood via adaptive Newton method. These methods are extremely popular since they are both computationally efficient and easy to implement. Moreover, the component-wise Boosting algorithms deal with the high dimensional issue by considering the explanatory one at a time. In each iteration, only the most effective explanatory variable is chosen to train a weak learner. Hence, these methods allow $k \gg n$. However, hyper-parameters such as the number of boosting iteration normally need to be determined by the user prior to the estimation procedure. Cross-validation may also be used to choose the number of iterations.

Next, we give an introduction to alternative methods such as Deep Neural

Network, Logistic Regression, and SIM-RODEO. Deep Neural Network is a kind of nonlinear statistical learning model features a network structure that is similar to the relationship between the neurons of the human brain. Deep Neural Network may be explained partly as a kind of basis transformation which leads to extreme flexibilities of the models. Deep Neural Network and its variants are the most popular prediction method at this time and are widely used in fields such as image and voice recognition.

Logistic Regression is a traditional method used intensively in economics for binary classification and probability prediction. Logistic Regression assumes that the probability that the output label is 1 conditional on x follows a logistic function of x. Under such assumption, the parameters of the model often have practical economic meaning, unlike machine learning methods that are often hard to interpret. However, Logistic Regression relies heavily on its parametric assumptions and is the least flexible model introduced in this chapter. In addition, to deal with the high-dimensional problem, we have to use the LASSO to control the number of explanatory variables chosen in the model.

SIM-RODEO relaxes the parametric assumption of Logistic Regression. As a result, SIM-RODEO is more flexible but, to some extent, still interpretable as Logistic Regression. However, the flexibility of SIM-RODEO may lead to a slower convergence rate and less time efficiency.

We conduct extensive comparisons of the above-mentioned methods through Monte Carlo experiments. We compare the methods using both traditional binary classification model (logistic model) and irregular model (circle model). The boosting methods work well in both the traditional models and irregular models. Logistic Regression works better in the low dimension logistic model when the parametric assumptions of Logistic Regression are satisfied. However, in the high-dimensional case, the LASSO introduces high bias in Logistic Regression and lead to lower classification

accuracy. In the irregular models, Logistic Regression performs poorly compared to the boosting algorithms. The Deep Neural Network performed best in the traditional methods as a result of our configuration of the Neural Network. We acknowledge that our configuration of Deep Neural Network is by no means the best and the results here may improve with different activation function, output function and/or the number of hidden layers and neurons. SIM-RODEO is an extension to parametric methods such as Logistic Regression. It performs reasonably well in the models. We also use these methods for predicting the changing direction of the real personal income in the United States. The application shows similar results as in the simulation of logistic models.

This chapter gives a thorough introduction of newly developed methods for binary classification and probability prediction. Advantages and disadvantages of each method are discussed and compared. We conclude that no single method has an absolute advantage in all aspects of the other methods. We believe binary classification and probability prediction will remain important for business and economics and look forward to future works on this problem.

Chapter 3

Asymmetric AdaBoost for High-dimensional Maximum Score Regression

3.1 Introduction

Data with a large number of variables relative to the sample size, namely high-dimensional data, are becoming more and more prevalent in empirical economics as well as statistics and computer science. One of the most successful applications of high-dimensional data in economics as well as other sciences is to construct empirical models for forecasting of binary outcomes and making binary decisions. Examples in forecasting include predicting firm solvency, the legitimacy of credit card transactions, directional forecasts of financial prices, whether a loan is paid off or not, or whether an introduced foreign plant species will become invasive or not. Such forecasts are often translated into decisions which are binary in character, e.g. the loan is granted or it is not, the student is admitted to the school or not, the candidate is hired

or not hired, the surgery is undertaken or it is not, importation of a foreign plant species is allowed or not. Various statistical approaches to binary classification are available in the literature, from discriminant analysis, logit or probit models to less parametric estimates of the conditional probability model for the outcome variable such as semiparametric single-index models (Ichimura, 1993; Klein and Spady, 1993).

Typically, most estimation techniques used for binary classification do not make use of the loss function implicit in the underlying decision/prediction problem. For example logit and probit models are estimated to maximize the likelihood of the model, irrespective of the relative usefulness of true positives or true negatives. Nonparametric methods seek the best fit for the conditional probability based on the loss function (typically squared error) rather than the appropriate loss function for the decision problem. In most applications, the relative costs of making errors, false negatives and false positives, are rarely balanced in the way that could be used to motivate these approaches. In detecting credit card fraud, "wasting" resources on checking that the customer has control over their credit card is perhaps less costly than failing to do so when their credit card number has been stolen. Elliott and Lieli (2013) point out that even with local misspecifications that are difficult to detect using standard specification tests, parametric models of the conditional probability of a positive outcome can perform arbitrarily poorly when the loss function is ignored at the estimation stage. They further propose the "maximum utility estimator" which is a semiparametric method that requires far less information to attain maximal utility, and through the utilization of the loss function at the estimation stage has useful properties given any misspecification. The maximum utility estimator builds upon and extends results of Manski (1985).

This chapter extends the method of Manski (1985) and Elliott and Lieli (2013) to high-dimensional data and model misspecification in the order of independent

variables. We consider the prediction of a binary variable $y \in \{1, -1\}$, e.g. $y = 1$ if the economy is in expansion and $y = -1$ if the economy is in recession. And let $G(x)$ be a classifier of y. This chapter investigates the problem of classification/prediction that minimizes a weighted (asymmetric) misclassification probability

$$\begin{aligned} R_\tau(G) &= \mathbb{E}\left[\tau(x) \times 1_{(y=-1,G(x)=1)} + (1-\tau(x)) \times 1_{(y=1,G(x)=-1)}\right] \quad (3.1) \\ &= \mathbb{E}_x[\tau(x) \Pr(y=-1, G(x)=1|x) + (1-\tau(x)) \Pr(y=1, G(x)=-1|x)], \quad (3.2) \end{aligned}$$

where the first expectation is taken over y and x, and the symbol $1_{(\cdot)}$ is the indicator function which takes the value 1 if the logical conditions inside the parenthesis are satisfied and takes the value 0 otherwise. $\tau(x)$ is a utility-based weight function that assigns different penalties conditioning on the state variable y and characteristics x as shown in Section 3.3. In addition, we allow the characteristics x to be high-dimensional, and both the conditional distribution of y given x and the functional form of the classifier $G(x)$ to be of unknown forms.

We propose a nonparametric method which minimizes an asymmetric exponential loss via functional gradient descent and builds a strong (optimal) classifier by iteratively combining weak classifiers. The resulted strong classifier can encamps a large class of functions even if the weak classifiers are restricted to a given parametric form. Moreover, we use component-wise algorithm and select only one independent variable at each iteration to overcome the issue of high-dimensionality.

There are some prediction problems that do not fit the framework examined here. A forecaster providing forecasts that might be used by a number of different users might not consider the loss function. For example a weather forecaster providing a forecast of whether or not it might rain might simply report an estimate of the conditional probability of rain and let different users interpret the information differently.

We will also rule out feedback of the prediction to the conditional probability of the event to be predicted, which means that the methods are not appropriate for predictions of outcomes where there is this type of feedback. Such feedback occurs for example in predicting success of job training programs, where entry to the program affects the chance of getting a job. However a myriad of problems are not ruled out, where the prediction is not important for the distribution of the outcomes and the econometrician is willing to elicit a loss function.

The rest of the chapter is organized as follows. In Section 3.4, we look into the problem of prediction with state-dependent losses and introduce a new "asymmetric exponential risk" function based on the utility functions. We also propose a new algorithm that minimizes the "asymmetric exponential risk" and builds up a nonparametric classifier. In Section 3.5, we examine the finite sample properties of Asymmetric AdaBoost via Monte Carlo simulations. Section 3.6 predicts business cycle turning points as in Ng (2014a). Section 3.7 concludes. All technical derivations and proofs are presented in the Appendix.

3.2 Binary Choice Model and Maximum Score

In this chapter, we consider the binary choice model given by

$$y = \begin{cases} 1 & \text{if } \phi(x) \geq \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

where $\phi(\cdot)$ is an unknown function, x is a vector of exogenous variables, ϵ is a random disturbance. We assume that observations $\{x_i, y_i\}$ are independently and identically distributed. However, we do not require any prior knowledge on the functional form

of $\phi(\cdot)$ or the distribution of ϵ.

Manski (1975) proposes to obtain a classifier by

$$G(x) = \arg\max_{G} \mathbb{E}\left[yG(x)\right], \tag{3.4}$$

which is called the maximum score approach. We maximize (3.4) with respect to $G(x) \in \{1, -1\}$,

$$\max_{G} \mathbb{E}\left[yG(x)|x\right] = \left[\Pr(y=1|x) - \Pr(y=-1|x)\right] G(x). \tag{3.5}$$

Hence, $G(x)$ takes the same sign as $\Pr(y=1|x) - \Pr(y=-1|x)$ when (3.4) is maximized, i.e.

$$G^*(x) = \begin{cases} 1 & \Pr(y=1|x) > \Pr(y=-1|x) \\ -1 & \text{otherwise,} \end{cases} \tag{3.6}$$

or equivalently,

$$G^*(x) = \begin{cases} 1 & \Pr(y=1|x) > 0.5 \\ -1 & \text{otherwise.} \end{cases} \tag{3.7}$$

Remark 3. We refer to the problem and the risk functions as "symmetric" since the optimal decision rule is $\Pr(y=1|x) > 0.5$, i.e. the optimal classifier uses 0.5 as the threshold. Similarly, we refer to the risk functions in Section 3.4 as "asymmetric" since the the optimal decision rule is not $\Pr(y=1|x) > 0.5$, i.e. the optimal classifier uses a threshold other than 0.5.

Note that the risk function (3.4) is a linear transformation of the misclassification

probability (3.1) with $\tau = 0.5$,

$$\mathbb{E}[-yG(x)] = 4 \times \mathbb{E}\left[\frac{1}{2} \times 1_{(G(x)\neq y)}\right] - 1 = 4 \times R_{0.5}(G) - 1. \tag{3.8}$$

Hence, the maximum score approach is equivalent to minimizing the symmetric misclassification probability.

From (3.7), the optimal maximum score classifier, also known as the Bayes classifier, makes classification based on the condition $\Pr(y = 1|x) > 0.5$. The Bayes classifier achieves the Bayes risk

$$R^* = \inf_G R(G) = \mathbb{E} \min\left\{\frac{1}{2}\Pr\left(y = 1|x\right), \frac{1}{2}\Pr\left(y = -1|x\right)\right\}, \tag{3.9}$$

where the infimum is taken over all possible (measurable) classifiers.

The maximum score approach yields a classifier that minimizes the misclassification probability (3.1) with $\tau = 0.5$. It is superior to many other popular methods, e.g. probit and logit models, in the sense that it does not have to assume that y given x follows a given distribution. However, there are some limitations: The classifier is assumed to take the form $G(x) = \text{sign}[x'\beta]$, i.e. the optimal classifier is the sign of a linear function; The objective function used is nonconvex which lead to computation difficulty especially when the sample size is large; The method does not work if covariates are high-dimensional.

3.3 Decision Theory for Binary Prediction/Classification

Granger and Pesaran (2000) discuss the idea of using decision theory to evaluate

classification/prediction accuracy in a two-state two-action decision problem. Assume the payoff matrix is

	$y=1$	$y=-1$
$G(x)=1$	$u_{1,1}(x)$	$u_{1,-1}(x)$
$G(x)=-1$	$u_{-1,1}(x)$	$u_{-1,-1}(x)$

(3.10)

where $u_{i,j}(x)$ is the state dependent utility of making prediction i when the realized value is j under circumstances x. Without loss of generality, we assume that $u_{1,1}(x) - u_{-1,1}(x) + u_{-1,-1}(x) - u_{1,-1}(x) = 1$. It is natural to also assume that all utilities are bounded and taking the correct decision i corresponding to realized state j is beneficial: $\tau(x) \equiv u_{1,1}(x) - u_{-1,1}(x) > 0$ and $1 - \tau(x) \equiv u_{-1,-1}(x) - u_{1,-1}(x) > 0$.

The expected utility of $G(x) = 1$ is

$$\Pr(y=1|x)\, u_{1,1}(x) + \Pr(y=-1|x)\, u_{1,-1}(x). \tag{3.11}$$

The expected utility of $G(x) = -1$ is

$$\Pr(y=1|x)\, u_{-1,1}(x) + \Pr(y=-1|x)\, u_{-1,-1}(x). \tag{3.12}$$

$G(x) = 1$ gives a higher utility if

$$\Pr(y=1|x)\, u_{1,1}(x) + \Pr(y=-1|x)\, u_{1,-1}(x) > \Pr(y=1|x)\, u_{-1,1}(x) + \Pr(y=-1|x)\, u_{-1,-1}(x). \tag{3.13}$$

Hence,

$$\Pr(y=1|x) > u_{-1,-1}(x) - u_{1,-1}(x) = 1 - \tau(x), \tag{3.14}$$

is the sufficient condition for $G(x) = 1$ to be the optimal choice.

The optimal decision rule depends only on $\tau(x) = u_{1,1}(x) - u_{-1,1}(x)$ and $1 - \tau(x) = u_{-1,-1}(x) - u_{1,-1}(x)$. Hence, without loss of generality, we can construct a problem with the same optimal classifier with $u'_{1,1}(x) = 0$, $u'_{-1,1}(x) = -\tau(x)$, $u'_{-1,-1}(x) = 0$ and $u'_{1,-1}(x) = -(1-\tau(x))$. Since the loss can be seen as the negative payoff, the constructed problem have a loss matrix as follows:

	$y=1$	$y=-1$
$G(x)=1$	0	$1-\tau(x)$
$G(x)=-1$	$\tau(x)$	0

(3.15)

where the risk can be summarized as (3.16). The state-dependent risk function is a weighted version of the score risk (Manski, 1985) as follows:

$$R_\tau(\text{sign}[F]) = \mathbb{E}\left(t(y,x)\,1_{(-y\,\text{sign}[F(x)]>0)}\right), \tag{3.16}$$

where

$$t(y,x) = \begin{cases} \tau(x) & y = 1 \\ 1-\tau(x) & y = -1. \end{cases} \tag{3.17}$$

is a non-negative function of outcome variable y and characteristics x. The score risk (3.16) is the counterpart of (3.1) with argument $F \in \mathbb{R}$ instead of $G \in \{1,-1\}$. Similarly, let

$$R^*_\tau = \inf_F R_\tau(\text{sign}[F]) = \mathbb{E}\{\min[t(1,x)\Pr(y=1|x), t(-1,x)\Pr(y=-1|x)]\} \tag{3.18}$$

be the Bayes risk which is the minimal risk can be achieved.

In this general case, the optimal classification rule is no longer $\Pr(y=1|x) > 0.5$.

The optimal classifier (also known as the Bayes classifier)

$$G_\tau^*(x) = \begin{cases} 1 & \Pr(y=1|x) > 1-\tau(x) \\ -1 & otherwise. \end{cases} \tag{3.19}$$

uses the classification rule (3.14) that is a function of the state-dependent utilities of the economic agent and achieves the Bayes risk (3.18).[1]

Remark 4. We refer to the binary classification/prediction problem with state-dependent losses as asymmetric since the optimal classification rule is $\Pr(y=1|x) > 1-\tau(x)$, i.e. the threshold is $1-\tau(x)$ instead of 0.5 as in the symmetric case.

3.4 Asymmetric Exponential Loss

In this section, we introduce a new risk function, namely the asymmetric exponential risk, for solving binary classification/prediction under state-dependent losses. We also propose a new algorithm, that we call the Asymmetric AdaBoost, which produces a nonparametric classifier by minimizing the asymmetric exponential risk. Our new algorithm is computationally efficient and is able to handle binary classification/prediction problem with high-dimensional covariates.

3.4.1 Convex Surrogate

The score risk (3.16) is nonconvex which lead to high computation cost especially when the sample size is large and/or covariates are high-dimensional. To solve the

[1]Manski (1975, 1985) propose the maximum score estimator to solve the above binary classification problem from minimizing a linear transformation of the score risk

$$\max_G E\left(t(y,x)yG\left(x\right)\right). \tag{3.20}$$

Elliott and Lieli (2013) also use a similar estimator which they call the maximum utility estimator.

problem, we propose to use a new risk function, the asymmetric exponential risk,

$$R_{\psi,\tau}(F) = \mathbb{E}\left(t(y,x)e^{-yF(x)}\right), \tag{3.21}$$

which is a convex surrogate of the score risk (3.16). Similarly, let us denote the optimal asymmetric exponential risk as

$$R^*_{\psi,\tau} = \inf_F R_{\psi,\tau}(F). \tag{3.22}$$

Similar to the previous sections, the asymmetric exponential risk replaces the nonconvex indicator function in the score risk (3.16) with the convex exponential function. As shown in Figure 3.1, the asymmetric exponential risk (3.21) is a convex upper bound of the score risk (3.16).

Note that the optimal classifier from minimizing the asymmetric exponential risk (3.21) also uses $\Pr(y=1|x) > 1-\tau(x)$ for the classification rule as in (3.19). Take the derivative of

$$\begin{aligned} R_{\psi,\tau}(F(x)) &= \mathbb{E}\left[\mathbb{E}\left(t(y,x)e^{-yF(x)}|x\right)\right] \\ &= \mathbb{E}\left[\tau(x)\Pr(y=1|x)e^{-F(x)} + (1-\tau(x))\Pr(y=-1|x)e^{F(x)}\right]. \end{aligned}$$

w.r.t. $F(x)$ and making it equal to zero, we obtain

$$\frac{\partial \mathbb{E}\left(e^{-yF(x)}|x\right)}{\partial F(x)} = -\tau(x)\Pr(y=1|x)e^{-F(x)} + (1-\tau(x))\Pr(y=-1|x)e^{F(x)} = 0. \tag{3.23}$$

Hence,

$$F_\tau^*(x) = \frac{1}{2}\log\left[\frac{\tau(x)\Pr(y=1|x)}{(1-\tau(x))\Pr(y=-1|x)}\right]. \tag{3.24}$$

Moreover, the optimal classifier,

$$G_\tau^*(x) = \text{sign}[F^*(x)] = \begin{cases} 1 & \Pr(y=1|x) > 1-\tau(x) \\ -1 & otherwise, \end{cases} \tag{3.25}$$

follows the classification rule $\Pr(y=1|x) > 1-\tau(x)$.

Theorem 1. *For every sequence of measurable functions $F_m : x \to \mathbb{R}$ and every probability distribution on $x \times \{\pm 1\}$,*

$$R_{\psi,\tau}(F_m) \to R_{\psi,\tau}^* \text{ implies that } R_\tau(\text{sign}[F_m]) \to R_\tau^*. \tag{3.26}$$

Proof. See Appendix A.1. □

Theorem 1 establishes the relationship between the convex asymmetric exponential risk and the nonconvex score risk that is widely used in decision theories such as the two-state two-action decision problem mentioned before. Therefore, we are able to replace the nonconvex risk function with a convex surrogate which could be minimized more efficiently and provide enormous improvement with large samples and high-dimensional data.

3.4.2 Asymmetric AdaBoost

In this section, we introduce our algorithm, which we call the Asymmetric AdaBoost, for minimizing our asymmetric exponential risk. We use functional gradient

descent to produce a nonparametric classifier. In addition, our algorithm can handle high-dimensional covariates. The algorithm is shown in Algorithm 6.

Algorithm 6 Asymmetric AdaBoost

1. Start with weights $w_i = t\left(y_i, x_i\right)$, $i = 1, \ldots, n$, and normalize so that $\sum_{i=1}^{N} w_i = 1$.
2. For $m = 1$ to M
   (a) For $j = 1$ to k (for each variable)
      i. Fit the classifier $f_{mj}(x_{ij}) \in \{-1, 1\}$ using weights w_i.
      ii. Compute $err_{mj} = \sum_{i=1}^{n} w_i 1_{(y_i \neq f_{mj}(x_{ij}))}$.
      iii. Compute $c_{mj} = \frac{1}{2} \log\left(\frac{1-err_{mj}}{err_{mj}}\right)$.
   (b) Find $\hat{j}_m = \arg\min_j \sum w_i e^{-c_{mj} y_i f_{mj}(x_{ij})}$.
   (c) Set $w_i \leftarrow w_i \exp[-c_{m\hat{j}} y_i f_{m\hat{j}}\left(x_{i\hat{j}}\right)]$, $i = 1, \ldots, n$, and normalize so that $\sum_{i=1}^{n} w_i = 1$.
3. Output the classifier, sign $\left[\sum_{m=1}^{M} c_m f_{m\hat{j}_m}(x_{\hat{j}_m})\right]$.

Remark 5. For the selection of the number of iterations M, a widely used method in the boosting literature is cross-validation. Here we can divide the whole sample into several sections, then take turns to use one section as test sample to evaluate the obtained model while using the other sections as training sample. In the end, we choose the number of iteration that has the least cross-validation loss. Another choice is to use information criterion, e.g. AICc. The exponential loss can be linked with log-likelihood of logistic models as in Ng (2014).

The Component-wise Asymmetric AdaBoost algorithm uses one explanatory variable at a time to fit a weak classifier $f_{mj}(x_j)$. In the end, the algorithm produces a strong classifier $F_M(x)$ by combining all the weak classifiers that uses different explanatory variables. Hence, the Component-wise Asymmetric AdaBoost overcomes

the high-dimensional data problem by selecting only one explanatory variable in each iteration and combining the weak classifiers across iterations. Moreover, the resulted strong classifier is a weighted sum of weak classifiers which is not required to satisfy any parametric assumption. To better understand the new algorithm, we follow the steps of Friedman et al. (2000b) to explain our Asymmetric AdaBoost.

Theorem 2. *Algorithm 6 builds an additive regression model $F_M(x)$ via Newton-like updates for minimizing the asymmetric exponential risk (3.21).*

Proof. See Appendix A.2. □

As in the previous sections, from the use of a convex risk function, Algorithm 2 is computationally more efficient. Moreover, since the convex exponential risk (**??**) is differentiable, Algorithm 2 uses functional gradient descent to minimize the asymmetric exponential risk which will produce a classifier with larger flexibility.

Theorem 3. *Let assumption 1 be satisfied. Then the Algorithm 2 stopped at iteration $M_n = n^{1-\epsilon}$ where $\epsilon \in (0,1)$ returns a sequence of classifiers F_{M_n} almost surely satisfying*

$$R_{\psi,\tau}(F_{M_n}) \to R^*_{\psi,\tau} \text{ as } n \to \infty. \tag{3.27}$$

Proof. See Appendix A.3. □

Theorem 3 shows that Algorithm 2 is consistent in the sense that the risk of the classifier obtained will converge to the optimal asymmetric exponential risk as the sample size goes to infinity, i.e. the classifier produced by Algorithm 2 will minimize the exponential risk (**??**). Moreover, by Theorem 1, the classifier will also achieve the Bayes risk (3.18).

Theorem 4. *Given the conditions in Theorem 3, the Algorithm 2 stopped at iteration* $M_n = n^{1-\epsilon}$ *where* $\epsilon \in (0,1)$ *returns a sequence of classifiers* F_{M_n} *almost surely satisfying*

$$R_\tau\left(\text{sign}[F_{M_n}]\right) \to R_\tau^* \text{ as } n \to \infty. \tag{3.28}$$

Proof. Combining Theorems 1 and 3 gives the above result. □

Algorithm 1 is a special case of Algorithm 2 when $t(y,x) = \frac{1}{2}$. Hence, Algorithm 2 is able to solve binary classification/prediction problem with state-dependent losses while maintaining the computation advantage and function form flexibility of Algorithm 1. In addition, Algorithm 2 can deal with high-dimensional x.

3.5 Monte Carlo

In this section, we examine the finite sample properties of the Asymmetric AdaBoost via Monte Carlo simulations and compare its performance with the Logistic Regression with LASSO-penalty. We consider the binary decision problem in Section 3.3 with $\tau(x) = \tau$.

3.5.1 DGPs

We construct the following high-dimensional DGPs where y follows Bernoulli distribution and x is high-dimensional. All the DGPs satisfy the sparsity assumption that most of the x's are completely irrelevant or have negligible influence on y.

DGP1 (Linear Logistic Models):

$$\Pr\left(y = 1|x\right) = \frac{1}{1+e^{-v}}.$$

Let x be a $p \times 1$ vector.

$$v = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_p x_p,$$

where

$$(x_1, x_2, \ldots, x_p)' \sim N(0, I_p), \quad \beta_j = 0.8^j, \ j = 1, \ldots, p$$
$$n = \{100, 1000\}, \quad p = 100.$$

DGP1 is the classical logistic model where the probability of y being 1 depends only on a single index v that is linear in x. This is the underlying model of the Logistic Regression. Hence, we would expect that Logistic Regression would be the best in DGP1. We construct DGP1 to give the most disadvantages to Asymmetric AdaBoost when comparing with Logistic Regression.

DGP2 (Quadratic Logistic Models):

$$\Pr(y = 1|x) = \frac{1}{1 + e^{-v}}.$$

Let x be a $p \times 1$ vector.

$$v = \beta_2(x_1^2 - x_2^2) + \beta_3 x_3 + \cdots + \beta_p x_p,$$

where

$$(x_1, x_2, \ldots, x_p)' \sim N(0, I_p), \quad \beta_j = 0.8^j, \ j = 2, \ldots, p$$
$$n = \{100, 1000\}, \quad p = 100.$$

DGP2 is a slight deviation from the classical logistic model in the sense that the single index v in the logistic model is not linear in x_1 and x_2. We take the difference of x_1^2 and x_2^2 so that the expectation of the single index v is 0 and the unconditional probability of $y = 1$ is 0.5, i.e., the data is balanced. We will examine the performance of the Asymmetric AdaBoost with unbalanced data in DGP4. Note that in the simulations, we provide the two methods with x of only the first order. Since the Asymmetric AdaBoost does not depend on any parametric assumptions, we would like to check the robustness of the Asymmetric AdaBoost and the sensitivity of the Logistic Regression when the model is slightly misspecified.

DGP3 (Cubic Logistic Models):

$$\Pr(y = 1|x) = \frac{1}{1 + e^{-v}}.$$

Let x be a $p \times 1$ vector.

$$v = x_1^3 - 4x_1,$$

where

$$(x_1, x_2, \ldots, x_p)' \sim N(0, I_p), \quad n = \{100, 1000\}, \quad p = 100.$$

In DGP3, we deviate further from the classical logistic model by having the single-index v to be a third-order polynomial of x_1. DGP3 is to test the performance of the Asymmetric AdaBoost and the Logistic Regression when the parametric assumptions of the Logistic Regression are invalid.

DGP4 (Circle Model, Mease, Wyner, and Buja (2007)):

$$\Pr(y=1|x) = \begin{cases} 1 & v < 8 \\ \frac{28-v}{20} & 8 \leq v \leq 28 \\ 0 & v > 28 \end{cases}.$$

Let x be a $p \times 1$ vector.

$$v = \sqrt{x_1^2 + x_2^2}$$

where

$$x_j \sim \mathrm{U}[-28, 28], \quad j = 1, \ldots, p$$
$$n = \{100, 1000\}, \quad p = 100.$$

The probability, $\Pr(y=1|x)$, in the DGP4 is shown in Figure 3.2. A major difference between DGP4 and the other DGPs is that $\Pr(y=1) \approx 0.1 < 0.5$ in DGP4. Hence, the data is unbalanced, i.e. there are more events of $y=-1$ than $y=1$. We have this setup since in many situations we are more interested in predicting an event that is less common than its complementary, e.g. recessions over expansions.

To construct the training and testing samples, we randomly generate x using the above distribution and calculate $\Pr(y=1|x)$. To generate the random variable y based on x, we first generate a random variable ϵ that follows uniform distribution between $[0,1]$. Next, we compare ϵ with $\Pr(y=1|x)$. There is a probability of $\Pr(y=1|x)$ that ϵ is smaller than $\Pr(y=1|x)$ and a probability $1 - \Pr(y=1|x)$

otherwise. Hence, we set

$$y = \begin{cases} 1 & \Pr(y = 1|x) > \epsilon \\ -1 & \Pr(y = 1|x) < \epsilon. \end{cases} \tag{3.29}$$

To evaluate the algorithms, first we train our classifier with the training data of size $n = \{100, 1000\}$. Then, we use a testing dataset that contains $n' = 10000$ new observations to test the out-of-sample performance of the methods.

We report the following sample version of the 0-1 risk of the tested methods,

$$\hat{R}_{\tau,n'}(\text{sign}[F]) = \frac{\tau}{n'} \sum_{y_i=1} 1_{(y_i \neq \text{sign}[F(x_i)])} + \frac{(1-\tau)}{n'} \sum_{y_i=-1} 1_{(y_i \neq \text{sign}[F(x_i)])}. \tag{3.30}$$

We also report the sample Bayes risk as the benchmark for comparison,

$$\hat{R}^*_{\tau,n'} = \frac{1}{n'} \sum_{i=1}^{n'} \min \{\tau \Pr(y = 1|x_i), (1-\tau) \Pr(y = -1|x_i)\}.$$

The above procedure is repeated for 1000 times and the average over the 1000 repetitions is reported in the tables.

3.5.2 Alternative Method: Asymmetric Logistic Regression

Apart from Asymmetric AdaBoost, we consider the Logistic Regression as an alternative method to obtain a classifier of y. In the alternative method, we use $Y = \frac{y+1}{2}$ for simplification. Because of the high-dimensional construction of our problem, we minimizes the negative logistic log-likelihood with a LASSO-penalty as below

$$\beta = \arg\min_{\beta} - \sum_{i=1}^{n} \left[Y_i (x_i \beta) - \log \left(1 + e^{x_i \beta}\right) \right] + \lambda |\beta|_1 . \tag{3.31}$$

In particular, we use the standard *glmnet* package of Friedman et al. (2010b) for the Logistic Regression. We use the estimated β to construct a logistic probability model for y. Then, get the classifications by plugging the estimated logistic probability into the Bayes classifier (3.19).

3.5.3 Results

The simulation results are reported in Tables 1 to 4. In Table 3.1, the DGP1 is a linear logistic model. In this case, the Logistic Regression has absolute advantage over Asymmetric AdaBoost both when n is small and large. This is expected since logistic regression has the correct parametric assumption in this case which is infeasible in practice. However, even in this case, we see that the advantage of the Logistic Regression over the Asymmetric AdaBoost is limited and as the sample size increases, the loss of the Asymmetric AdaBoost converges to the sample Bayes risk which suggests that the Asymmetric AdaBoost is consistent.

In Table 2, the DGP2 is still the logit model. Hence, the Logistic Regression still has inherited advantages over the Asymmetric AdaBoost. However, we introduce a small deviation from DGP1 by letting the single index, v, in the logistic function be quadratic in x_1 and x_2. In this case, the logistic regression is partially biased since it assumes that the single index is a linear function of the covariates. When n is small, we see that the results are neck and neck. The Asymmetric AdaBoost works better when τ is close to 0.5 and the logistic regression works better when τ is away from 0.5. This is expected as our method is nonparametric and nonparametric methods generally perform worse in the tails when samples in the area are few. Moreover, Logistic Regression is also not highly biased. Both methods are far behind the Bayes risk since the Asymmetric AdaBoost without parametric assumption has larger variance

and the logistic regression with wrong parametric assumption is biased.

When the sample size increases, the Asymmetric AdaBoost have smaller variance and the losses are closer to the sample Bayes risk. The Logistic Regression, on the other hand, is still biased and has higher losses than the Asymmetric AdaBoost except in the two far tails. This shows that the Asymmetric AdaBoost that produces a nonparametric classifier will suffer from higher variance if the sample size is small. But, as the sample size increases, the Asymmetric AdaBoost will produce an unbiased classifier and achieve lower losses than logistic regression which is biased even if the true model only deviates slightly from the parametric assumptions of the Logistic Regression.

In Table 3, the DGP3 deviates further from the classical logistic model. The Asymmetric AdaBoost performs strictly better than the Logistic Regression. When n is small, we see that the Asymmetric AdaBoost outperforms the Logistic Regression except in the two tails ($\tau = 0.1$ and $\tau = 0.9$) where insufficient samples are available. This is a general limitation of all nonparametric methods since nonparametric methods. However, the performance of the Asymmetric AdaBoost surpasses the Logistic Regression in the tails when the sample size become larger. In practice, when the true DGP is not the logistic model, the Asymmetric AdaBoost is definitely more reliable.

In Table 4, the DGP4 is unbalanced. The event $y = 1$ is significantly fewer than $y = -1$. We can see that the Asymmetric AdaBoost works better when the minority of the events is penalized more heavily. The Asymmetric AdaBoost has lower losses on the right-hand side where $y = 1$ is penalized more heavily, and higher losses on the left-hand side where $y = -1$ is penalized more heavily. In the unbalanced DGP, the Logistic Regression only focuses on the event that is the majority. However, the Asymmetric AdaBoost still tries to model both events. Hence, if one is interested in predicting the less common event, e.g. recession over expansion, the Asymmetric

AdaBoost will give lower losses as we will see in the application section. Moreover, as the sample size increases, we see that the Asymmetric AdaBoost converges to the Bayes risk on both sides and catches up with logistic regression on the left-hand side.

In summary, the Asymmetric AdaBoost is consistent in the sense that the losses of the classifier produced converges to the sample Bayes risk as the sample size increases. Compared with the Logistic Regression, the Asymmetric AdaBoost is more robust if the true DGP is not the logistic model especially when the sample size if large. Moreover, the Asymmetric AdaBoost is better than the Logistic Regression if one is more interested in predicting the less common events, such as recessions over expansions, when the data is unbalanced.

3.6 Application

In this section, we predict the NBER business cycle turning points using both the Asymmetric AdaBoost and the Logistic Regression with LASSO-penalty. We use the 132 independent variables from the data of Jurado, Ludvigson, and Ng (2015). After removing the observations with missing values and taking the one, two and three lagged values of each independent variable and the dependent variable, the remained sample period ranges from April 1964 to July 2011 with 568 observations and 399 independent variables. We use a rolling sample scheme and make three-period ahead predictions of economic recessions as in Ng (2014b). We use rolling sample size (n) of 60, 120 and 240. The average losses from all rolling samples under different degrees of asymmetry

$$\hat{R}_{\tau,n}\left(\text{sign}[F]\right) = \frac{\tau}{n'}\sum_{y_i=1} 1_{(y_i \neq \text{sign}[F(x_i)])} + \frac{(1-\tau)}{n'}\sum_{y_i=-1} 1_{(y_i \neq \text{sign}[F(x_i)])}$$

where n' is the total number of rolling samples are reported in Table 3.5.

In the application, we see that the Asymmetric AdaBoost has smaller losses than the Logistic Regression. Both the Asymmetric AdaBoost and the Logistic Regression are consistent in the sense that the forecasting error decreases as the rolling sample size increases. Algorithm-wise, we have removed the rolling samples that contain less than two months of recessions. These samples account for 119, 2, 0 of the total rolling samples in the cases where the rolling sample sizes are 60, 120 and 240. When the number of recessions in the rolling sample is less than two, the standard package for Logistic Regression with LASSO-penalty reports an error and fails to produce the result(Friedman et al., 2010b). More specifically, if there is no recession contained in the rolling sample, the maximum likelihood of the Logistic Regression would be 0, i.e., the coefficients of the Logistic Regression would explode to infinity. In addition, we can not use cross-validation to choose the penalty term λ for Logistic Regression since the cross-validation process involves randomly resampling the rolling samples and frequently results in less than two recessions in the cross-validation samples even when $n = 240$. Instead, we tried different values of λ for Logistic Regression and reported all of them in Table 5. In almost all cases, the Asymmetric AdaBoost significantly outperforms the Logistic Regression which strongly suggests that the parametric assumptions of Logistic Regression are invalid in this application.

3.7 Conclusions

In this chapter, we introduce a new Asymmetric AdaBoost algorithm which produces an additive regression model from maximizing a new risk function, namely the asymmetric exponential risk function. The new Asymmetric AdaBoost algorithm is based on the asymmetric exponential risk function, which maps into a binary

decision making problem given a utility function. Furthermore, by carefully establishing the asymmetry in the risk function in accordance to the binary decision making, we show that our Asymmetric AdaBoost algorithm is closely related to the maximum score regression (Manski 1975, 1985) and the binary prediction literature in economics (Granger and Pesaran 2000, Lee and Yang 2006, Lahiri and Yang 2012, and Elliot and Lieli 2013), all of which however deal with low-dimensional predictor space. Asymmetric AdaBoost can handle the maximum score and binary prediction when the predictors are high-dimensional. Theoretical results show that Asymmetric AdaBoost will converge to Bayes risk as $n \to \infty$. Simulation and application results show that Asymmetric AdaBoost is a competitive approach in binary classification/prediction.

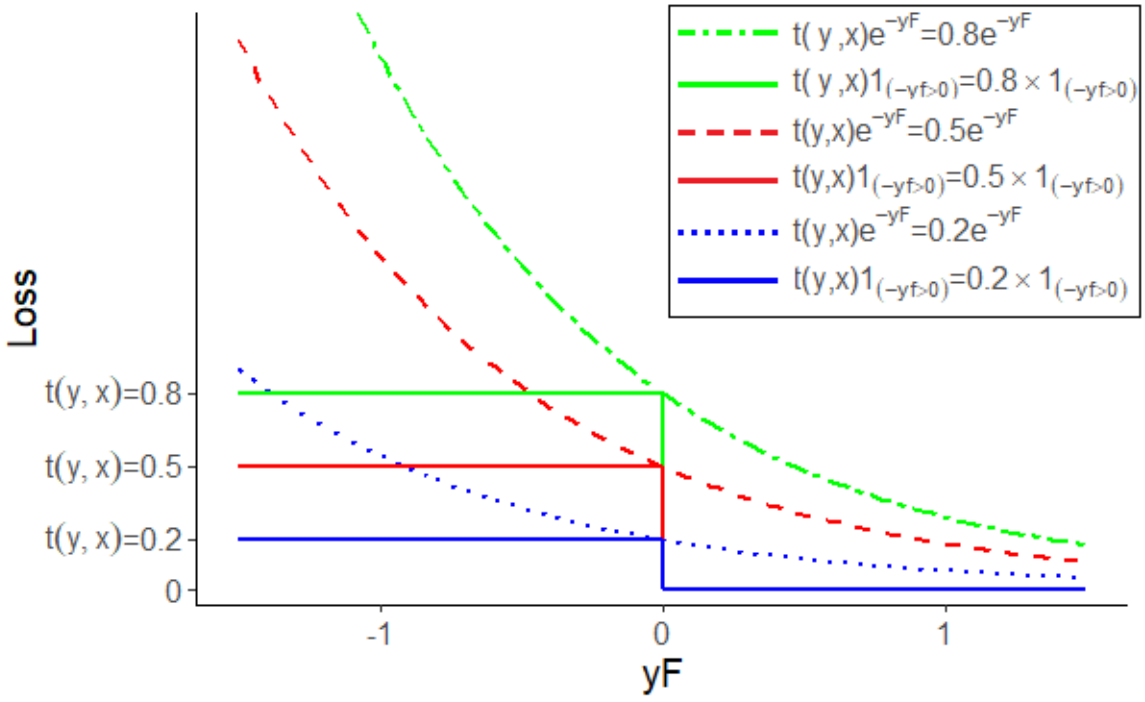


Figure 3.1: (Asymmetric) Exponential Loss and Score Loss

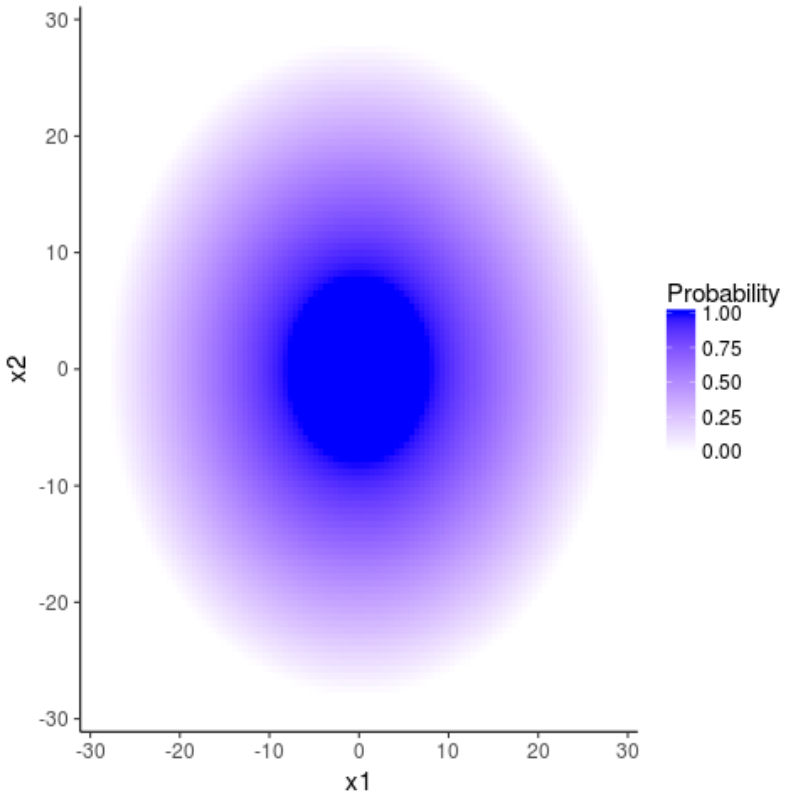


Figure 3.2: Conditional Probability of the Circle Model

Table 3.1: Linear Logit Model (DGP1)

	τ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	AdaBoost	0.0544	0.0997	0.1379	0.1602	0.1712	0.1607	0.1372	0.1005	0.0545
$n = 1000$	LASSO	0.0492	0.0934	0.1266	0.1479	0.1550	0.1482	0.1271	0.0933	0.0493
	Bayes Risk	0.0482	0.0885	0.1178	0.1360	0.1419	0.1359	0.1182	0.0886	0.0482
	AdaBoost	0.0774	0.1263	0.1728	0.2001	0.2085	0.1981	0.1739	0.1300	0.0773
$n = 100$	LASSO	0.0509	0.1026	0.1483	0.1814	0.1973	0.1843	0.1482	0.1015	0.0513
	Bayes Risk	0.0482	0.0885	0.1180	0.1357	0.1418	0.1358	0.1179	0.0885	0.0483

Note: The average of the losses of the two methods for predicting y are reported in the table. Bayes Risk is the infeasible optimal risk when the true model is known. τ shows different degrees of asymmetry. n is the sample size of each training sample.

Table 3.2: Balanced Quadratic Logit Model (DGP2)

	τ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
	AdaBoost	0.0524	0.0958	0.1330	0.1614	0.1736	0.1570	0.1316	0.0951	0.0516
$n = 1000$	LASSO	0.0510	0.1021	0.1495	0.1841	0.1949	0.1805	0.1442	0.0977	0.0488
	Bayes Risk	0.0469	0.0866	0.1168	0.1358	0.1422	0.1358	0.1170	0.0867	0.0469
	AdaBoost	0.0765	0.1304	0.1734	0.2017	0.2121	0.2049	0.1769	0.1335	0.0819
$n = 100$	LASSO	0.0501	0.1017	0.1552	0.2076	0.2346	0.2063	0.1541	0.1030	0.0514
	Bayes Risk	0.0468	0.0866	0.1168	0.1357	0.1422	0.1358	0.1168	0.0866	0.0469

Note: The average of the losses of the two methods for predicting y are reported in the table. Bayes Risk is the infeasible optimal risk when the true model is known. τ shows different degrees of asymmetry. n is the sample size of each training sample.

Table 3.3: Unbalanced Quadratic Logit Model (DGP3)

	τ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$n = 1000$	AdaBoost	0.0442	0.0642	0.0770	0.0837	0.0857	0.0837	0.0771	0.0641	0.0443
	LASSO	0.0500	0.0999	0.1499	0.1999	0.2499	0.1998	0.1499	0.1000	0.0500
	Bayes Risk	0.0402	0.0609	0.0736	0.0807	0.0830	0.0807	0.0736	0.0609	0.0402
$n = 100$	AdaBoost	0.0539	0.0843	0.1148	0.1393	0.1393	0.1392	0.1162	0.0843	0.0535
	LASSO	0.0500	0.0999	0.1500	0.2015	0.2498	0.2023	0.1501	0.0999	0.0500
	Bayes Risk	0.0403	0.0609	0.0736	0.0807	0.0830	0.0807	0.0737	0.0609	0.0402

Note: The average of the losses of the two methods for predicting y are reported in the table. Bayes Risk is the infeasible optimal risk when the true model is known. τ shows different degrees of asymmetry. n is the sample size of each training sample.

Table 3.4: Circle Model (DGP4)

	τ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$n = 1000$	AdaBoost	0.0402	0.0719	0.0835	0.0893	0.0981	0.1041	0.1058	0.0794	0.0443
	LASSO	0.0358	0.0715	0.1073	0.1430	0.1792	0.2158	0.1937	0.1283	0.0641
	Bayes Risk	0.0276	0.0513	0.0700	0.0833	0.0902	0.0897	0.0814	0.0640	0.0372
$n = 100$	AdaBoost	0.0554	0.0841	0.1090	0.1256	0.1353	0.1387	0.1336	0.1189	0.0807
	LASSO	0.0358	0.0718	0.1082	0.1451	0.1848	0.2272	0.2049	0.1344	0.0658
	Bayes Risk	0.0276	0.0512	0.0700	0.0834	0.0902	0.0897	0.0812	0.0640	0.0372

Note: The average of the losses of the two methods for predicting y are reported in the table. Bayes Risk is the infeasible optimal risk when the true model is known. τ shows different degrees of asymmetry. n is the sample size of each training sample.

Table 3.5: Loss for Predicting Recessions

	τ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$n = 60$	AdaBoost	0.0181	0.0255	0.0318	0.0377	0.0374	0.0405	0.0431	0.0362	0.0348
	LASSO ($\lambda = 0.05$)	0.0229	0.0350	0.0452	0.0499	0.0514	0.0473	0.0422	0.0365	0.0288
	LASSO ($\lambda = 0.1$)	0.0216	0.0370	0.0522	0.0535	0.0488	0.0473	0.0458	0.0411	0.0440
	LASSO ($\lambda = 1$)	0.0211	0.0422	0.0632	0.0843	0.1054	0.1326	0.1707	0.1769	0.1262
	LASSO ($\lambda = 5$)	0.0211	0.0422	0.0632	0.0843	0.1054	0.1326	0.1707	0.1769	0.1262
$n = 120$	AdaBoost	0.0154	0.0245	0.0267	0.0316	0.0334	0.0352	0.0348	0.0276	0.0223
	LASSO ($\lambda = 0.05$)	0.0182	0.0314	0.0370	0.0381	0.0381	0.1641	0.1269	0.0919	0.0511
	LASSO ($\lambda = 0.1$)	0.0168	0.0336	0.0469	0.0444	0.0471	0.0457	0.0365	0.0359	0.0332
	LASSO ($\lambda = 1$)	0.0155	0.0309	0.0464	0.0619	0.0774	0.0928	0.1128	0.1439	0.0659
	LASSO ($\lambda = 5$)	0.0155	0.0309	0.0464	0.0619	0.0774	0.0928	0.1128	0.1439	0.0659
$n = 240$	AdaBoost	0.0115	0.0158	0.0228	0.0237	0.0289	0.0256	0.0271	0.0231	0.0170
	LASSO ($\lambda = 0.05$)	0.0112	0.0213	0.0295	0.0402	0.0442	0.1121	0.0871	0.0621	0.0378
	LASSO ($\lambda = 0.1$)	0.0112	0.0225	0.0338	0.0426	0.0472	0.0408	0.0274	0.0274	0.0320
	LASSO ($\lambda = 1$)	0.0112	0.0225	0.0338	0.0451	0.0564	0.0676	0.0789	0.1500	0.0853
	LASSO ($\lambda = 5$)	0.0112	0.0225	0.0338	0.0451	0.0564	0.0676	0.0789	0.1500	0.0853

Note: The average losses of the three period ahead prediction on recessions are reported in the table. τ shows different degrees of asymmetry.

Chapter 4

Variable Selection in Sparse Semiparametric Single-index Models

4.1 Introduction

This chapter proposes a new method of variable selection for sparse semiparametric single-index models, that would be useful for semiparametric Probit and Logit models with many regressors. Nadaraya (1964) and Watson (1964) propose the Nadaraya-Watson local constant kernel regression estimator. Kernel regression has been extremely popular for it is free of parametric assumptions. On the other hand, it suffers from computational complexity and the curse of dimensionality. Ichimura (1993) studies the semiparametric single index model (SIM) to overcome the curse of dimensionality by assuming that the true model is a function of an index which is a linear combination of the explanatory variables. Klein and Spady (1993) study a similar semiparametric single index model for binary outcomes and propose to estimate the model by maximum

likelihood. However, these SIM methods gain limited improvements computationally over the local constant and local linear kernel regression and are still slow to implement.

Recent statistics and econometrics literature has been focusing on big data issues which are extremely difficult to solve with kernel regressions. To overcome this problem, under the sparsity assumption, several papers propose regularized SIM methods with penalty terms. See e.g. Huang et al. (2010). Su and Zhang (2014) provide a comprehensive review on those literature. However, those penalties may induce additional complexity in computation and lead to huge bias and variance when the ratio of information to noise is small. One such method that seems to be a natural way for SIM is a LASSO-type approach by Zeng et al. (2012) for estimation and variable selection in SIM, which they termed as "SIM-LASSO".

Meanwhile, there is a large volume of literature motivated by statistical machine learning, such as AdaBoost, Boosting, Support Vector Machine, Deep neural net. In particular, in this chapter, we note that the method of Lafferty and Wasserman (2008b), called the Regularization of Derivative Expectation Operator (RODEO), may be modified for SIM. RODEO is a greedy algorithm for variable selection and estimation of the nonparametric regression function based on testing of marginal contribution of an additional variable in selecting relevant explanatory variables. A goal of this chapter is to modify RODEO so that it can be applied to semiparametric SIMs under sparsity. We will call the modified RODEO for SIM as "SIM-RODEO".

The SIM-RODEO method is able to distinguish relevant explanatory variables from irrelevant variables and gives a competitive estimator for the model. In addition, the algorithm finishes in a reasonable period of time. The method assumes sparsity under which most of the explanatory variables are irrelevant. We use a greedy algorithm that starts with a semiparametric SIM estimator (Ichimura, 1993) that

sets all coefficients $\left(\theta_j = \frac{\beta_j}{h}\right)$ as zero which are the ratio of slope coefficients β_j to bandwidth h in the original Ichimura estimator. Then, we iteratively test if the derivative of the regression estimator with respect to each coefficient θ_j is zero. The intuition is for a relevant explanatory variable, changing its coefficient would lead to a dramatic change in the value of the estimator. However, for an irrelevant variable, changing its coefficient would lead to ideally no change to the single index estimator. The impact of changing the coefficient to the attained estimator can be measured with the derivative of the estimator with respect to the coefficient. If the derivative with respect to one coefficient is zero, it implies the corresponding explanatory variable does not have a strong explanatory power on the dependent variable. And it will be seen as an irrelevant variable and given coefficient zero. However, if the derivative with respect to one coefficient is significantly different from zero, then we say the corresponding explanatory variable has a strong explanatory power on the dependent variable. Hence, it will be seen as a relevant explanatory variable and given coefficient greater than zero. The proposed procedure attains a solution path similar to the Least Angle Regression (Efron and Hastie, 2004). The new method is superior to the usual LASSO type penalty (Zeng et al., 2012) in the sense that it does not introduce bias into the estimation process, is free of user-specific parameters and computationally more efficient. Simulation results show that the proposed method is consistent for variable selection and has smaller integrated mean squared errors (IMSE) than using LASSO penalty.

The rest of the chapter is organized as follows. Section 4.2 introduces the intuition and algorithm of the original RODEO of Lafferty and Wasserman (2008b). Section 4.3 sets up a model for the semiparametric single index model of Ichimura (1993), introduces the SIM-RODEO, and discusses the asymptotic properties of SIM-RODEO in variable selection and estimation of the semiparametric single index model.

Section 4.4 provides Monte Carlo simulation results for SIM-RODEO in comparison with SIM-LASSO of Zeng et al. (2012). Section 4.5 concludes.

4.2 RODEO

This section introduces the idea behind the Regularization of Derivative Expectation Operator (RODEO) proposed by Lafferty and Wasserman (2008b). We first provide an illustration of the RODEO algorithm, and then, a simple numerical example with one relevant explanatory variable and one irrelevant noise variable.

4.2.1 Algorithm

Let $y_i \in \mathbb{R}$ be the dependent variable, $X_i \in \mathbb{R}^k$ be an observation of k variables, $X = (X_1', \ldots, X_n')'$ be a matrix of n observations and $x \in \mathbb{R}^k$ be a local estimation point.

The RODEO algorithm uses the kernel estimator

$$\hat{m}_h(x) = \frac{\sum_{i=1}^n y_i K(X_i, x, h)}{\sum_{i=1}^n K(X_i, x, h)}, \tag{4.1}$$

where h is a vector of length k that is equal to the number of potential explanatory variable, h_j is the jth element of h that is corresponding to variable j and $K(X_i, x, h)$ is the standard notation of a product kernel that takes the form

$$K(X_i, x, h) = \prod_{j=1}^{k} \kappa\left(\frac{X_{ij} - x_j}{h_j}\right), \tag{4.2}$$

where $\kappa(\cdot)$ is usually given as a one-variable density function. X_{ij} is the ith observation of the jth variable and x_j is the jth variable of a local estimation point x. In what

follows, we keep the same notation except that in the single index model, our kernel becomes a one-variable density function instead of a product kernel.

The RODEO algorithm takes the derivative of the kernel estimator (4.1) with respect to each bandwidth h_j. Let

$$\iota = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \tag{4.3}$$

and

$$W_x = \begin{pmatrix} K(X_1, x, h) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K(X_n, x, h) \end{pmatrix}. \tag{4.4}$$

With fairly easy derivation, we can get the closed form of an estimate of the derivative

$$\begin{aligned} Z_j &\equiv \frac{\partial \hat{m}_h(x)}{\partial h_j} \\ &= (\iota' W_x \iota)^{-1} \iota' \frac{\partial W_x}{\partial h_j} y - (\iota' W_x \iota)^{-1} \iota' \frac{\partial W_x}{\partial h_j} \iota (\iota' W_x \iota)^{-1} \iota' W_x y \\ &\equiv \sum_{i=1}^{n} G_j(X_i, x, h)\, y_i, \end{aligned} \tag{4.5}$$

where $y = (y_1, \ldots, y_n)$ is a vector of observations on the dependent variable. The conditional variance of Z_j can be calculated by

$$\begin{aligned} s_j^2 &\equiv \mathrm{Var}(Z_j | X) \\ &= \sigma^2 \sum_{i=1}^{n} G_j^2(X_i, x, h), \end{aligned} \tag{4.6}$$

Algorithm 7 RODEO (Lafferty and Wasserman, 2008b)

1. Select constant $0 < \alpha < 1$ and initial bandwidth

$$h_0 = \frac{c_0}{\log \log n}$$

   where $c_0 > 0$ is sufficiently large.

2. Initialize the bandwidths, and activate all covariates:
   (a) $h_j = h_0$, $j = 1, \ldots, k$.
   (b) $\mathcal{A} = \{1, \ldots, k\}$.

3. While $\mathcal{A}$ is nonempty, do for each $j \in \mathcal{A}$:
   (a) Compute the estimated derivative and its conditional variance: Z_j and s_j using (4.5) and (4.6).
   (b) Compute the threshold $\lambda_j = s_j \sqrt{2 \log n}$.
   (c) If $|Z_j| > \lambda_j$, then set $h_j \leftarrow \alpha h_j$; otherwise remove j from $\mathcal{A}$ (i.e., $\mathcal{A} \leftarrow \mathcal{A} - \{j\}$).

4. Output bandwidths $h^* = (h_1, \ldots, h_k)$ and estimator $\hat{m}_{h^*}(x)$ where $\hat{m}_{h^*}(x)$ is the kernel estimator with bandwidth h^*.

where the detailed derivation can be found in Section 3 of Lafferty and Wasserman (2008b). Here we skip the derivation for the kernel regression. However, we provide a detailed derivation for the single index model (SIM) in Section 4. Now we get all the ingredients of the RODEO algorithm. The RODEO algorithm is as follows.

The basic idea of the RODEO algorithm by (Lafferty and Wasserman, 2008b) is to view the local bandwidth selection as variable selection in sparse nonparametric kernel regression models by shrinking the bandwidths for relevant variables while keeping the bandwidths of irrelevant variables relatively large. The RODEO algorithm is greedy as it solves for the locally optimal path choice at each iteration and is shown to attain the consistency in mean square error when it is applied to sparse

nonparametric local linear model (Lafferty and Wasserman, 2008b, Corollary 5.2).[1]

4.2.2 A Numerical Example

Now we give a numerical illustration of how RODEO works. First we generate 100 data points from the DGP

$$y = \frac{1}{1 + e^{-x_1}} + u, \tag{4.7}$$

where x_1 is a random variable following uniform distribution with range $[-3, 3]$ and u is a random variable following the normal distribution with mean 0 and standard deviation 0.02. The generated data of x_1 and y are shown in Figure 4.1.

Next, we generate an irrelevant variable x_2 that follows the same distribution as x_1 but is not included in the model. Thus, x_2 and y are independent. The generated data of x_2 and y are shown in Figure 4.2.

In the algorithm, we start by setting bandwidths h_j for all j large enough so that

$$\frac{X_{ij} - x_j}{h_j} \to 0 \quad \text{for all } i. \tag{4.8}$$

Hence,

$$K(X_i, x, h) \to \prod_{j=1}^{k} \kappa(0). \tag{4.9}$$

For simplicity of illustration, we assume the kernel function is an indicator function $\kappa(X_{ij}, x_j, h_j) = 1(|X_{ij} - x_j| < h_j)$. This makes our estimate a simple average of the observations that satisfy $|X_{ij} - x_j| < h_j$ for all j. If for all j, $|X_{ij} - x_j|$ is smaller than the bandwidth h_j, then we include observation i in the average. Otherwise, we

[1] We can also make RODEO for local constant kernel regression models as we will demonstrate in this chapter later.

exclude it. At the beginning, when the bandwidths are large enough, our estimate is the global mean since all observations are included in the estimate. However, if we shrink the bandwidth h_j, we exclude the observations whose X_{ij} has a distance greater than h_j from x_j. Hence, our estimate change from the global mean $\hat{m}_h(x) = \bar{y}$ to a local mean $\hat{m}_h(x)$ as shown in Figure 4.3.

The local mean in general has a smaller bias since taking the Taylor expansion, we have

$$m(x_i) = m(x) + \frac{\partial m(x)}{\partial x_1}(x_{i1} - x_1) + o(x_{i1} - x_1),$$

which shows that smaller distance between x_{i1} and x_1 implies smaller distance between $m(x_i)$ and $m(x)$ since

$$\frac{\partial m(x)}{\partial x_1} \neq 0.$$

Hence, giving more weights to observations that are closer to our local estimate point in x_1 would yield a better estimate than the global mean. However, this holds only for x_1. Shrinking the bandwidth of x_2 does not have the same effect. Since x_2 is not included in the true model, it does not appear in the Taylor expansion of $m(\cdot)$ or equivalently

$$\frac{\partial m(x)}{\partial x_2} = 0.$$

Closer in x_2 does not implies closer in $m(\cdot)$. In fact, from Figure 4.4 we can see that shrinking the bandwidth of x_2 does not affect the value of the estimate. This observation gives us a criteria to distinguish between relevant explanatory variables and irrelevant variables.

4.3 RODEO for Single Index Model (SIM-RODEO)

In this section, we show that RODEO can be modified for the sparse semiparametric linear single index models by considering the bandwidths as the inverse of the parameters which form the linear single index.

First, we give a short introduction to the general set up of the SIM model and the Ichimura (1993) estimator we use for estimation. We also give detailed intuition and description of our proposed greedy estimation procedure.

4.3.1 SIM-Model

We consider a standard single index model,

$$y = m\left(x'\beta\right) + u, \tag{4.10}$$

where $\beta = (\beta_1, \ldots, \beta_k)$ is a vector of coefficients. Under the sparsity condition, we assume that $\beta_j \neq 0$ for $j \leq r$ and $\beta_j = 0$ for $j > r$. We also assume that the random errors u are independent. However, we allow the presence of heteroskedasticity to encompass a large category of models for binary prediction, e.g. Logit and Probit models. The kernel estimator (Ichimura, 1993) we use is as shown below

$$\hat{m}\left(x'\beta; h\right) = \frac{\sum_{i=1}^{n} y_i K\left(\frac{X_i'\beta - x'\beta}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i'\beta - x'\beta}{h}\right)}, \tag{4.11}$$

where $K\left(\cdot\right)$ is a kernel function. The semiparametric kernel regression looks for the best β and h to minimize a weighted squared error loss. However, exact identification is not available. If one blows up β and θ simultaneously by multiplying the same constant, the kernel estimator would yield identical estimates and losses. The standard

identification approach is to set the first element of β to be 1 (Ichimura, 1993).

As recent research pays more attention to high-dimensional data, most literature makes the sparsity assumption that many, if not most, of the elements of β are zero. The previous mentioned identification method appears to be unsuitable unless we have specific information that the true value of the element of β that we set to be 1 is not zero. The most popular regularization method, LASSO (Tibshirani, 1996), also fails for the same reason. With L_1 penalty, the algorithm can always achieve a lower loss by shrinking β and h while keeping the ratio of $\frac{\beta}{h}$ constant. This would lead to a lower value in the penalty term without changing the value of the squared error term.

In terms of variable selection and prediction, we only need to focus on finding the best $\theta \equiv \frac{\beta}{h}$. Hence, we can simplify the estimator to

$$\hat{m}\left(x'\theta\right) = \frac{\sum_{i=1}^{n} y_i K\left(X_i'\theta - x'\theta\right)}{\sum_{i=1}^{n} K\left(X_i'\theta - x'\theta\right)}. \tag{4.12}$$

Instead of the standard two stage estimation of Ichimura (1993), we introduce a test-based greedy approach similar to Lafferty and Wasserman (2008b) where it was used for bandwidth selection in local linear regression. The intuition for the method is that if x_j is a relevant explanatory variable of y, then we would expect that increasing the magnitude of θ_j would lead to a significant change in $\hat{m}\left(x'\theta\right)$. This can be seen as giving higher weights to the observations closer to $x'\theta$ and lower weights to the observations further away from $x'\theta$. On the other hand, if x_j is not a relevant explanatory variable of y, then increasing the magnitude of θ_j can be seen as randomly reassigning weights for the observations and will only result in a random (moderate) change in $\hat{m}\left(x'\theta\right)$. The influence of changing the magnitude of θ_j on $\hat{m}\left(x'\theta\right)$ can be measured as the derivative of $\frac{\partial \hat{m}(x'\theta)}{\partial \theta_j}$. Hence, we can test if x_j is a

relevant explanatory variable by testing if $\frac{\partial \hat{m}(x'\theta)}{\partial \theta_j}$ is statistically different from zero.

4.3.2 SIM-RODEO

The basic idea of the modified RODEO algorithm for SIM (SIM-RODEO) is to view the local bandwidth selection as a variable selection in sparse semiparametric single index model. The SIM-RODEO algorithm amplifies the inverse of the bandwidths for relevant variables while keeping the inverse of the bandwidths of irrelevant variables relatively small. The SIM-RODEO algorithm is greedy as it solves for the locally optimal path choice at each iteration. It can also be shown to attain the consistency in mean square error when it is applied for sparse semiparametric single index models. SIM-RODEO is able to distinguish truly relevant explanatory variables from noisy irrelevant variables and gives a consistent estimator of the regression function. In addition, the algorithm is fast to finish the greedy steps.

Now we derive the RODEO for Single Index Models. First we introduce some notation. Let

$$W_x = \begin{pmatrix} K\left(X_1'\theta - x'\theta\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K\left(X_n'\theta - x'\theta\right) \end{pmatrix} \tag{4.13}$$

where $K(\cdot)$ is the Gaussian kernel. The standard Ichimura (1993) estimator takes the form

$$\hat{m}\left(x'\theta\right) = \frac{\sum_{i=1}^{n} y_i K\left(X_i'\theta - x'\theta\right)}{\sum_{i=1}^{n} K\left(X_i'\theta - x'\theta\right)} = \left(\iota' W_x \iota\right)^{-1} \iota' W_x y. \tag{4.14}$$

The derivative of the estimator Z_j with respect to θ_j is

$$\begin{aligned} Z_j &\equiv \frac{\partial \hat{m}\left(x'\theta\right)}{\partial \theta_j} && (4.15) \\ &= \left(\iota' W_x \iota\right)^{-1} \iota' \frac{\partial W_x}{\partial \theta_j} y - \left(\iota' W_x \iota\right)^{-1} \iota' \frac{\partial W_x}{\partial \theta_j} \iota \left(\iota' W_x \iota\right)^{-1} \iota' W_x y \\ &= \left(\iota' W_x \iota\right)^{-1} \iota' \frac{\partial W_x}{\partial \theta_j} \left(y - \iota \hat{m}\left(x'\theta\right)\right). && (4.16) \end{aligned}$$

For the ease of computation, let

$$L_j = \begin{pmatrix} \frac{\partial \log K\left(X_1'\theta - x'\theta\right)}{\partial \theta_j} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial \log K\left(X_n'\theta - x'\theta\right)}{\partial \theta_j} \end{pmatrix}. \tag{4.17}$$

Note that

$$\frac{\partial W_x}{\partial \theta_j} = W_x L_j, \tag{4.18}$$

which appears in equation (4.16). With the Gaussian kernel, $K\left(t\right) = e^{-\frac{t^2}{2}}$, then L_j becomes

$$\begin{aligned} L_j &= \begin{pmatrix} -\frac{1}{2}\frac{\partial \left(X_1'\theta - x'\theta\right)^2}{\partial \theta_j} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -\frac{1}{2}\frac{\partial \left(X_n'\theta - x'\theta\right)^2}{\partial \theta_j} \end{pmatrix} \\ &= \begin{pmatrix} -\left(X_1'\theta - x'\theta\right)\left(X_{1j} - x_j\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -\left(X_n'\theta - x'\theta\right)\left(X_{nj} - x_j\right) \end{pmatrix}, \end{aligned}$$

where X_{1j} and X_{nj} are the jth elements of vectors X_1 and X_n. And x_j is the jth element of vector x. To simplify the notation, let $B_x = \left(\iota' W_x \iota\right)^{-1} \iota' W_x$. Then, the

derivative Z_j becomes

$$\begin{aligned} Z_j &= (\iota' W_x \iota)^{-1} \iota' \frac{\partial W_x}{\partial \theta_j} (y - \iota \hat{m}(x'\theta)) \\ &= B_x L_j (I - \iota B_x) y \\ &\equiv G_j(x, \theta) y. \end{aligned} \tag{4.19}$$

Note that now we are using a different notation with $G_j(\cdot)$. In Section 2, $G_j(\cdot)$ is a three-argument function and $G_j(X_i, x, h)$ is a scalar. However, in this section, $G_j(\cdot)$ is a two-argument function and $G_j(x, \theta)$ is a vector of length n. We are aware that this change of notation may cause confusion. Nevertheless, $G_j(\cdot)$ in Sections 4.2 and 4.3 play the same role as the weights of y in Z_j. So we think sticking with $G_j(\cdot)$ would be easier for the readers to understand and compare RODEO and SIM-RODEO as long as the difference is pointed out and noticed by the readers.

Next, we give the conditional expectation and variance of Z_j.

$$Z_j = G_j(x, \theta) y = G_j(x, \theta)(m(x'\beta) + u), \tag{4.20}$$

$$E(Z_j|X) = E(G_j(x, \theta)(m(x'\beta) + u)|X) = G_j(x, \theta) m(x'\beta), \tag{4.21}$$

$$\mathrm{Var}(Z_j|X) = \mathrm{Var}(G_j(x, \theta)(m(x'\beta) + u)|X) = \boldsymbol{\sigma}' G_j(x, \theta)' G_j(x, \theta) \boldsymbol{\sigma}, \tag{4.22}$$

where $\boldsymbol{\sigma} = (\sigma(u_1), \ldots, \sigma(u_n))'$ is the vector of standard deviations of u. In the algorithm, it is necessary to insert an estimate of σ. In Algorithm 7, Lafferty and Wasserman (2008b) suggest to use a generalized estimator of Rice (1984) under homoskedasticity. In our Algorithm 8, we allow the errors to be heteroskedastic as in Logit and Probit models and estimate $\sigma(u_i)$ using the estimator $\hat{\sigma}(u_i) = m(x_i'\hat{\theta})(1 - m(x_i'\hat{\theta}))$. SIM-RODEO is described in Algorithm 2, which is a modified

Algorithm 8 SIM-RODEO

1. Select a constant $0 < \alpha < 1$ and the initial value

$$\theta_0 = c_0 \log\log n$$

   where c_0 is sufficiently small. Compute Z_j with $\theta_j = \theta_0$ for all j.

2. Initialize the coefficients θ, and activate all covariates:

   (a) $\theta_j = \begin{cases} \theta_0 & Z_j > 0 \\ -\theta_0 & \text{otherwise,} \end{cases} \quad j = 1, \ldots, k.$

   (b) $\mathcal{A} = \{1, \ldots, k\}$.

3. While $\mathcal{A} \neq \emptyset$ is nonempty, do for each $j \in \mathcal{A}$:

   (a) Compute Z_j and $s_j = \sqrt{\mathrm{Var}\left(Z_j | X\right)}$ using (4.19) and (4.22) respectively.

   (b) Compute the threshold $\lambda_j = s_j \sqrt{2 \log n}$.

   (c) If $|Z_j| > \lambda_j$, then set $\theta_j \leftarrow \frac{\theta_j}{\alpha}$; Otherwise, remove j from $\mathcal{A}$ (i.e., $\mathcal{A} \leftarrow \mathcal{A} - \{j\}$).

4. Output $\hat{\theta} = (\theta_1, \ldots, \theta_k)$ and estimator $\hat{m}\left(x'\hat{\theta}\right)$.

algorithm of RODEO (Lafferty and Wasserman, 2008b).

Notice that in Algorithm 7, when selecting bandwidth for local linear and local constant regression, the bandwidth is always positive. Hence, we do not have to worry about the sign of the bandwidth. However, in our single index model, θ is the ratio of β and the bandwidth. Since β could be either positive or negative, θ could also take positive or negative values. In Algorithm 8, we propose to use the sign of the derivative estimate Z_j as the sign of θ_j. Our method is based on the observation that if θ_j and $\theta_{j'}$ have the same sign, then their respective Z statistic Z_j and $Z_{j'}$ will also have the same sign. Hence, SIM-RODEO will give relatively correct signs to each θ, i.e. all the positive θ will be given the same sign and all the negative θ will be given the same sign. A similar method is applied by Ichimura (1993) where the value positive one is given to the first β to ensure identification. Under the sparsity assumption, it is problematic to arbitrarily assign a magnitude greater than zero to any θ since the true value could be zero. However, it is safe to assume the sign of one of the θ to be positive or negative since positive zero and negative zero will not affect the relative scale of θ. Once again, due to the the identification issue with single index model, exact identification of θ is not available. However, signs of θ can be obtained relatively.

We start by setting $\theta_j = \theta_0$ that is close to zero. Hence, $(X_i'\theta - x'\theta)$ are close to zero and $K\left(X_i'\theta - x'\theta\right)$ are close to $K\left(0\right)$. This means our estimator starts with the simple average of all observations, $\bar{y}$. If the derivative of θ_j is statistically different from zero. We amplify θ_j. If x_j is indeed a relevant explanatory variable, then the weights $K\left(X_i'\theta - x'\theta\right)$ change according to x_j. The estimator will give higher weights to observations close to $x'\theta$ and lower weights to observations away from $x'\theta$.

4.3.3 Asymptotic Properties of SIM-RODEO

We make the following assumptions.

Theorem 1. *(A1) The density $f(x)$ of $(x_1, \ldots, x_k)$ is uniform on the unit cube.*

(A2) $\liminf_{n\to\infty} \min_{1\le j\le r} |m_{jj}(\cdot)| > 0$ *where $m_{jj}(\cdot)$ is the second derivative of $m(\cdot)$.*

(A3) All derivatives of $m(\cdot)$ up to and including fourth order are bounded.

Remarks. Assumption (A1) greatly simplifies the proof of theorem 1. However, it is not necessary as shown in our Monte Carlo designs where x's are not uniform distributed. Assumption (A2) is crucial for SIM-RODEO. As we seen in Lemma 1, the expectation of Z_j for a relevant variable will be zero if the second derivative of $m(\cdot)$ is zero. As a result, we will not be able to distinguish relevant variables from irrelevant variables through Z_j since in both cases, the expectation of Z_j is zero.

In the statement of Theorem 1, we follow the notation of (Lafferty and Wasserman, 2008b) and write $Y_n = \tilde{O}(a_n)$ to mean that $Y_n = O(b_n a_n)$ where b_n is logarithmic in n. And we write $a_n = \Omega(b_n)$ if $\liminf_n \left|\frac{a_n}{b_n}\right| > 0$ and $a_n = \tilde{\Omega}(b_n)$ if $a_n = \Omega(b_n c_n)$ where c_n is logarithmic in n.

Theorem 5. *Suppose that Assumptions (A1), (A2) and (A3) hold. In addition, suppose that*

$$\min_{j\le r} |m_{jj}(x'\theta)| = \tilde{\Omega}(1) \tag{4.23}$$

and

$$\max_{j\le r} |m_{jj}(x'\theta)| = \tilde{O}(1). \tag{4.24}$$

Then the SIM-RODEO outputs $\hat{\theta}$ satisfying

$$\Pr(\theta_j = \theta_0 \text{ for all } j > r) \to 1 \text{ as } n \to \infty \tag{4.25}$$

and

$$\Pr\left(\theta_j > \theta_0 \text{ for all } j \leq r\right) \to 1 \text{ as } n \to \infty. \tag{4.26}$$

Proof. See Appendix. □

Theorem 1 shows that under the given assumptions and conditions, the coefficients θ for relevant variables will always be amplified while the coefficients θ for irrelevant variables will always stay at the initial value. Hence, we are able to consistently select the relevant variables by checking whether the coefficients θ is amplified by SIM-RODEO.

4.4 Monte Carlo

This section examines the performance of SIM-RODEO using Monte Carlo simulation compared with SIM-LASSO (Zeng et al., 2012) and Maximum Likelihood (Klein and Spady, 1993). We first descripe the designs of the DGPs. Then a breif introduction of SIM-LASSO is provided. At the end of this section, we give a comprehensive discussion on the simulation results.

4.4.1 Simulation Designs

We follow the simulation designs of Klein and Spady (1993) where the data generating process (DGP) is given by

$$y_i^* = \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots + \beta_k x_{ik} + u_i \quad \text{for} \quad i = 1, \ldots, n \tag{4.27}$$

where

$$\beta_j = \begin{cases} 1 & \text{if } j = 1, 2; \\ 0 & \text{otherwise.} \end{cases} \tag{4.28}$$

The observed variable y_i is generated by

$$y_i = \begin{cases} 1 & \text{if } y_i^* \geq 0; \\ 0 & \text{otherwise.} \end{cases} \tag{4.29}$$

The x's are independently and identically distributed. x_1 is a chi-squared variate with 3 degrees of freedom truncated at 6 and standardized to have zero mean and unit variance; x_2 is a standard normal variate truncated at ± 2 and similarly standardized. All the other x's are irrelevant variables and follow uniform distribution between -2 and 2.

We consider two link functions as in Figure 4.5 (Design 1 and Design 2). In Design 1, the u_i's are standard normal. In Design 2, they are normal with mean zero and variance $0.25(1 + v_i^2)^2$ where $v_i \equiv \beta_1 x_{i1} + \beta_2 x_{i2}$. In both designs, u_i's are independently distributed.

The probability $\Pr(y = 1|v)$ of the two designs are shown in Figure 4.5. Design 1 is the standard probit model. Design 2 is different from Design 1 in the sense that it is not monotone and is steeper in the tails. Hence, Design 2 has a larger curvature than Design 1 on average. As a result, SIM-RODEO is expected to preform better under Design 2 since assumption (A2) and conditions (4.23) and (4.24) requires the second derivative of the link function to be greater than zero.

4.4.2 SIM-LASSO

We show results for SIM-RODEO together with SIM-LASSO (Zeng et al., 2012) to check the relative efficiency of SIM-RODEO. The SIM-LASSO is introduced as an application of the LASSO penalty under the framework of Semiparametric Single Index Models for variable selection and esitmation. Zeng et al. (2012) propose to solve the following minimization problem

$$\min_{a,b,\beta,\|\beta\|=1} \sum_{j=1}^{n}\sum_{i=1}^{n} \left[y_i - a_j - b_j\beta'\left(X_i - X_j\right)\right]^2 w_{ij} + \lambda \sum_{j=1}^{n} |b_j| \sum_{p=1}^{k} |\beta_p| \tag{4.30}$$

where λ is a hyper parameter as in standard LASSO practices and

$$w_{ij} = \frac{K\left(\frac{X_i'\beta - X_j'\beta}{h}\right)}{\sum_{q=1}^{n} K\left(\frac{X_q'\beta - X_j'\beta}{h}\right)}. \tag{4.31}$$

The authors are generous enough to provide their code in the supplemental materials of their paper which is available on the website of *Journal of Computational and Graphical Statistics*.

4.4.3 Results

We report $\theta\left(=\frac{\beta}{h}\right)$ from SIM-RODEO and SIM-LASSO both for the estimator of Ichimura (1993). The results of the simulations are presented in Tables 1-4. Notice that for both algorithms, large values of θ indicate that the associated variables are relevant explanatory variables while small values of θ indicate that the associated variable are irrelevant variables. In both designs, only the first two variables are relevant explanatory variables as in the description of the DGPs. We consider different values for $n \in \{100, 200\}$ and $k \in \{5, 20\}$ where n is the number of observations in

the training sample and k is the total number of relevant explanatory variables and irrelevant variables for each observation. We also present results using the maximum likelihood (ML) of Klein and Spady (1993) for the low dimension case ($k = 5$). We skip the ML for the high dimension case ($k = 20$) since maximum likelihood suffers dramatically from the curse of dimensionality. The maximum likelihood is strictly dominated by the other methods even in the low dimension case. And theoretically, it would only get worse when dimension increases. We report the Monte Carlo average of the value of $\hat{\theta}$ obtained by the methods and the integrated mean squared error

$$\text{IMSE} = \int \left(\hat{m}(x'\hat{\theta}) - m(x'\theta)\right)^2 f(x)\mathrm{d}x \tag{4.32}$$

of the estimate $\hat{m}\left(x'\hat{\theta}\right)$ using the $\hat{\theta}$ obtained where $f(x)$ is the probability density function of x as in assumption (A1).

From the simulation results, we can see that under the sparsity condition, SIM-RODEO and SIM-LASSO both outperform the traditional maximum likelihood method of Klein and Spady (1993) which does not take advantage of the sparsity structure in the DGP. While among the two methods that take into account the sparsity structure, SIM-RODEO outperforms SIM-LASSO in both variable selection and estimation. In Design 2, SIM-RODEO dominates SIM-LASSO in small and large samples and various degrees of sparsity. In addition, SIM-RODEO works better under Design 2 than Design 1. This is consistent with our analytical result since the expectation of the derivative estimate Z_j is depending on the second derivative of $m(\cdot)$. When the second derivative of $m(\cdot)$ is close to zero, the expectations of Z_j of relevant explanatory variables are also close to zero which makes the difference between relevant variables and irrelevant variabls smaller. Moreover, the conditions (4.23) and (4.24) in Theorem 1 states that SIM-RODEO requires a larger value for the second derivative of $m(\cdot)$

when the number of observations n increases. As a result, when n increases from 100 to 200, the already small second derivative in Design 1 becomes even more problematic. That is why in Table 1, the IMSE of SIM-RODEO for Design 1 does not benefit from the increase of sample size. In summary, SIM-RODEO and SIM-LASSO both have excellent performance in terms of variable selection. However, SIM-RODEO generally has a smaller IMSE than SIM-LASSO. Maximum likelihood should not be used when sparsity is assumed since both SIM-RODEO and SIM-LASSO have considerably better performance.

4.5 Conclusions

The basic idea of the RODEO algorithm by Lafferty and Wasserman (2008b) is to view the local bandwidth selection as variable selection in sparse nonparametric kernel regression by shrinking the bandwidths for relevant variables while keeping the bandwidths of irrelevant variables relatively large. The RODEO algorithm is greedy as it solves the locally optimal path choice at each stage which is shown to attain the asymptotic optimality in mean square error for sparse nonparametric local linear or local constant kernel regression models Lafferty and Wasserman (2008b, Corollary 5.2).

In this chapter, we propose a new algorithm, based on the RODEO, for variable selection and estimation for the sparse semiparametric linear single index models by viewing the bandwidths as the inverse of the parameters which form the linear single index. The basic idea of the modified RODEO algorithm for SIM (which we call SIM-RODEO) is to view the local bandwidth selection as a variable selection in sparse semiparametric single index model by amplifying the inverse of the bandwidths for relevant variables while keeping the inverse of the bandwidths of irrelevant variables

relatively small. The SIM-RODEO algorithm is greedy as it solves the locally optimal path choice at each stage which can also be shown to attain the asymptotic optimality in mean square error for sparse semiparametric single index models. The SIM-RODEO method is able to distinguish truly relevant explanatory variables from noisy irrelevant variables and gives a "competitive" estimator for the model. In addition, the algorithm is fast to finish the greedy steps.

We compare the SIM-RODEO with a LASSO-type approach by Zeng et al. (2012) for estimation and variable selection in SIM, which Zeng et al. (2012) call SIM-LASSO. Our Monte Carlo simulation shows that SIM-RODEO outperforms SIM-LASSO in variable selection and also in estimation. The new method is superior to the usual LASSO type penalty in estimation because SIM-RODEO does not introduce bias from using the additive LASSO penalty and is computationally more efficient. Simulation results also show that the proposed SIM-RODEO is consistent for variable selection and has smaller integrated mean squared errors than using SIM-LASSO.

Table 4.1: Design 1 ($k = 5$)

		θ_1	θ_2	θ_3	θ_4	θ_5	IMSE of $m(x\theta)$
$n = 100$	RODEO	0.5739	0.3422	0.0713	0.0693	0.0724	0.0774
	LASSO	0.6032	0.5822	0.0223	0.0228	0.0235	0.1136
	ML	15.7902	11.3961	2.2156	2.2339	2.2334	0.3103
$n = 200$	RODEO	0.8063	0.5095	0.1811	0.1894	0.1904	0.0780
	LASSO	0.6572	0.6348	0.0142	0.0141	0.0142	0.0740
	ML	13.6022	11.6887	1.5990	1.6322	1.6093	0.2356

Table 4.2: Design 2 ($k = 5$)

		θ_1	θ_2	θ_3	θ_4	θ_5	IMSE of $m(x\theta)$
$n = 100$	RODEO	0.2486	0.1452	0.0160	0.0120	0.0057	0.0474
	LASSO	0.5241	0.4919	0.0332	0.0334	0.0357	0.1137
	ML	10.0696	4.5824	1.5773	1.6003	1.5993	0.1858
$n = 200$	RODEO	0.5022	0.2803	0.0296	0.0393	0.0426	0.0369
	LASSO	0.6547	0.6031	0.0209	0.0192	0.0207	0.0616
	ML	7.8625	4.7588	0.8896	0.8920	0.9034	0.1321

Table 4.3: Design 1 ($k = 20$)

		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	IMSE
$n = 100$	RODEO	0.2349	0.2117	0.0032	0.0048	0.0027	0.0049	0.0013	0.0080	0.0009	0.0023	0.1487
		θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}	θ_{17}	θ_{18}	θ_{19}	θ_{20}	
	RODEO	0.0093	0.0066	0.0037	0.0045	0.0026	0.0036	0.0016	0.0033	0.0030	0.0016	
		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	IMSE
	LASSO	0.4140	0.3904	0.0036	0.0036	0.0044	0.0045	0.0046	0.0048	0.0047	0.0047	0.2105
		θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}	θ_{17}	θ_{18}	θ_{19}	θ_{20}	
	LASSO	0.0063	0.0041	0.0056	0.0047	0.0048	0.0057	0.0034	0.0047	0.0045	0.0048	
$n = 200$		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	IMSE
	RODEO	0.4191	0.3404	0.0036	0.0104	0.0182	0.0120	0.0109	0.0118	0.0115	0.0060	0.1238
		θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}	θ_{17}	θ_{18}	θ_{19}	θ_{20}	
	RODEO	0.0077	0.0123	0.0086	0.0082	0.0074	0.0147	0.0105	0.0156	0.0105	0.0081	
		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	IMSE
	LASSO	0.4308	0.4120	0.0021	0.0029	0.0026	0.0024	0.0023	0.0023	0.0018	0.0019	0.1572
		θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}	θ_{17}	θ_{18}	θ_{19}	θ_{20}	
	LASSO	0.0021	0.0025	0.0026	0.0026	0.0020	0.0027	0.0022	0.0026	0.0025	0.0025	

Table 4.4: Design 2 ($k = 20$)

$n = 100$		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	IMSE
	RODEO	0.0692	0.0479	0.0007	0.0011	0.0000	0.0007	0.0000	0.0001	0.0004	0.0001	0.0611
		θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}	θ_{17}	θ_{18}	θ_{19}	θ_{20}	
	RODEO	0.0004	0.0001	0.0000	0.0000	0.0000	0.0007	0.0000	0.0000	0.0002	0.0000	
		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	IMSE
	LASSO	0.3488	0.2933	0.0104	0.0120	0.0121	0.0111	0.0106	0.0117	0.0106	0.0091	0.2078
		θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}	θ_{17}	θ_{18}	θ_{19}	θ_{20}	
	LASSO	0.0094	0.0109	0.0113	0.0093	0.0117	0.0120	0.0123	0.0112	0.0107	0.0106	
$n = 200$		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	IMSE
	RODEO	0.1958	0.1822	0.0018	0.0010	0.0024	0.0017	0.0008	0.0026	0.0022	0.0025	0.0517
		θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}	θ_{17}	θ_{18}	θ_{19}	θ_{20}	
	RODEO	0.0014	0.0024	0.0024	0.0002	0.0003	0.0041	0.0005	0.0013	0.0036	0.0029	
		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	IMSE
	LASSO	0.4324	0.3810	0.0058	0.0069	0.0066	0.0064	0.0059	0.0059	0.0063	0.0055	0.1728
		θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}	θ_{17}	θ_{18}	θ_{19}	θ_{20}	
	LASSO	0.0058	0.0057	0.0053	0.0054	0.0067	0.0069	0.0063	0.0072	0.0052	0.0049	

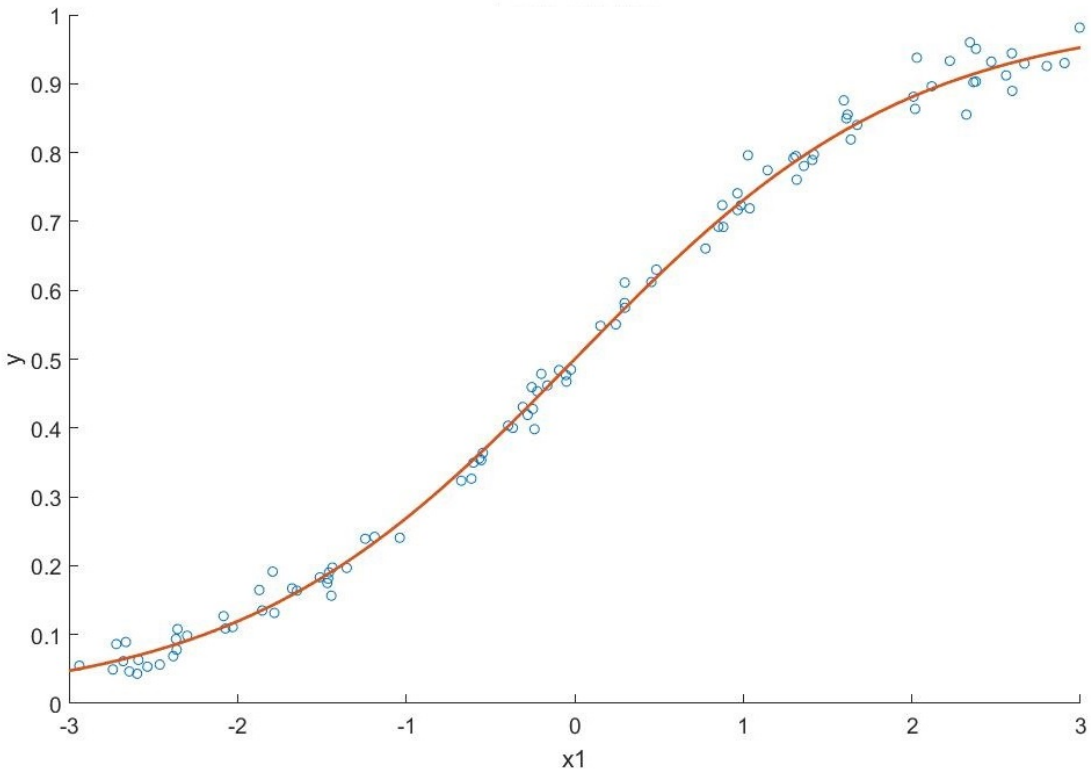


Figure 4.1: y with x_1

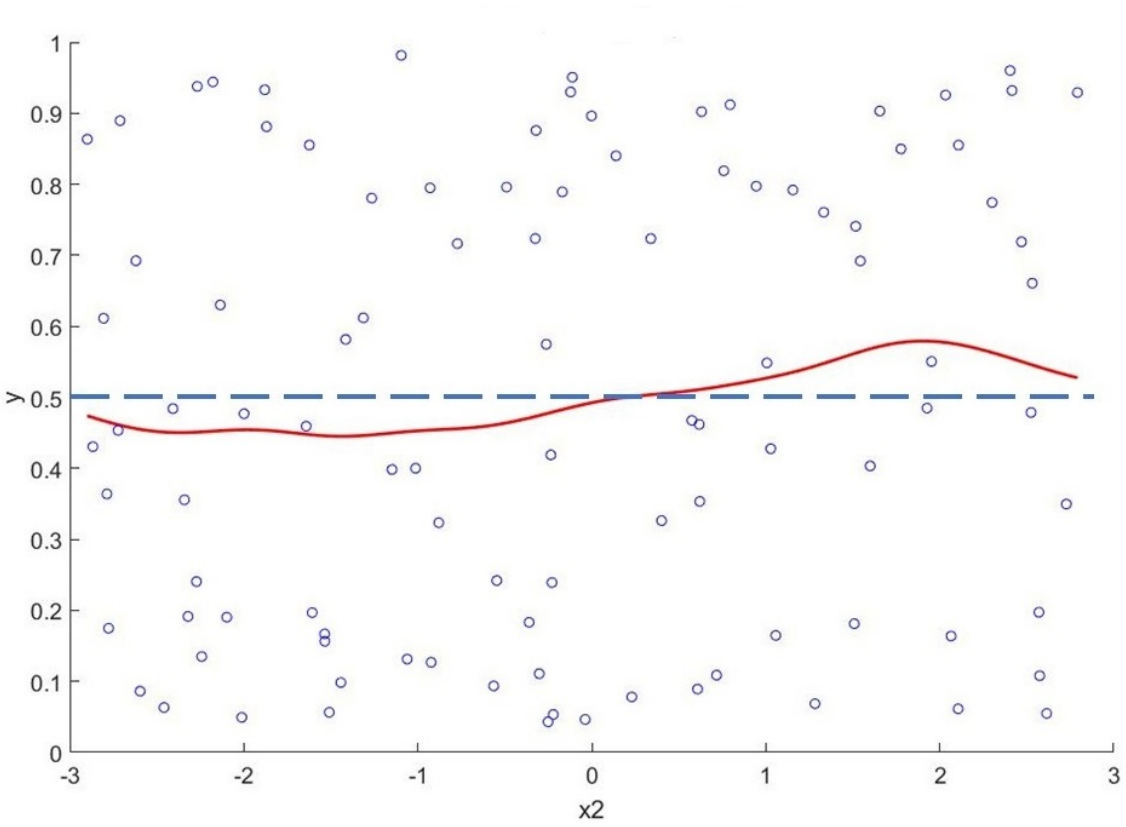


Figure 4.2: y with x_2

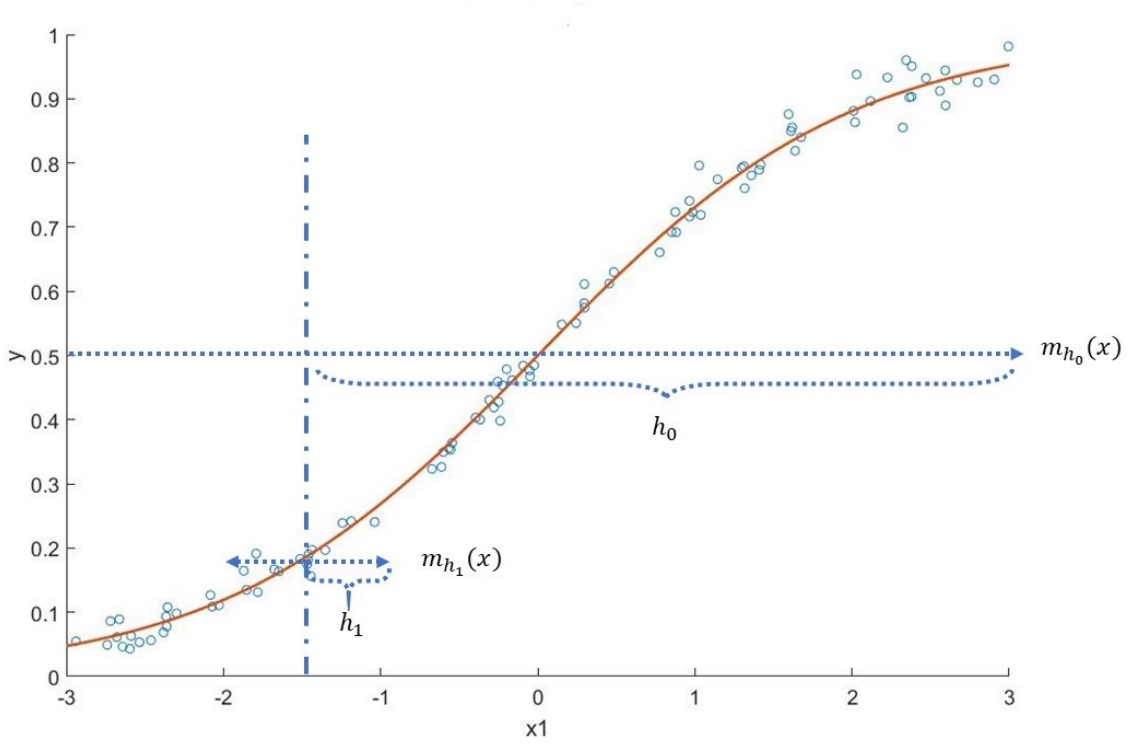


Figure 4.3: Shrink bandwidth of x_1 from h_0 to h_1

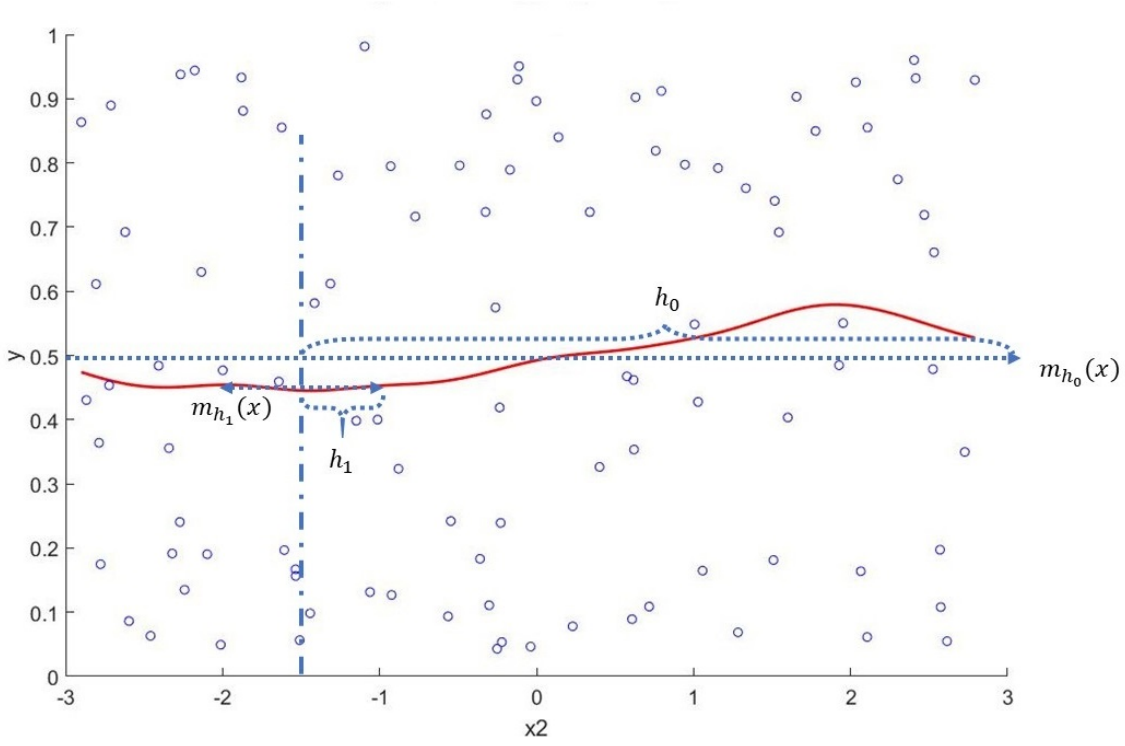


Figure 4.4: Shrinking bandwidth of x_2 from h_0 to h_1

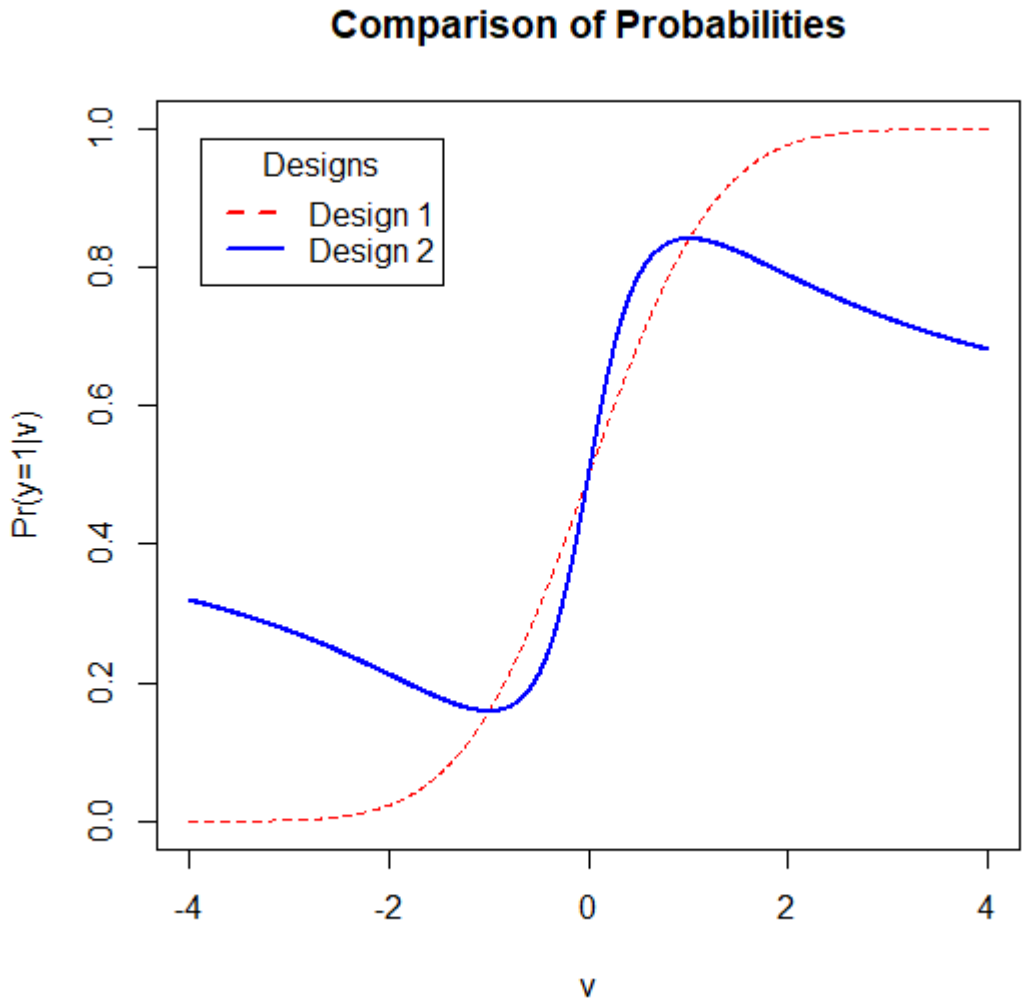


Figure 4.5: Designs

Chapter 5

Statistical Inference under Heteroskedasticity of Unknown Form

5.1 Introduction

The presence of heteroskedasticity of unknown forms in the disturbances of linear regression models has been a topic of major concern for inference in economic models. Under heteroskedasticity, the ordinary least squares (OLS) estimators of the regression parameters are well known to be consistent. However, the usual estimators for the covariance matrix of the regression parameters are inconsistent and/or biased. Numerous work has been done to formulate an unbiased and consistent estimate for the covariance matrix of the regression parameters. Rao (1970) proposes the Minimum Norm Quadratic Unbiased Estimation of the heteroskedastic variances. Other contributions include Eicker (1963), Huber (1967) and Hinkley (1977). White (1980), in a seminal paper, introduces the well-known "White Standard Errors" which

is able to estimate the covariance matrix of the regression parameters consistently under an unknown form of heteroskedasticity of the disturbances.

Not long after White (1980) established the asymptotic consistency of the heteroskedasticity-consistent covariance estimator, it was quickly recognized that inferences based on the "White Standard Errors" over-reject when the null hypothesis is true and the sample is not large. MacKinnon and White (1985b) provide simulation evidence and, linking the problem to the reduction in error variance brought by least squares fitting, further propose variants of the "White Standard Errors" to achieve better finite sample properties through adjusting for degrees of freedom and leverages and by using the jackknife. These corrections, however, are far from satisfactory, as shown in simulations by Angrist and Pishke (2009), who note that the covariance estimates are not merely biased but also much more variable than default OLS estimates, which contributes to their high rejection rates. Extensions on inferences using variants of the heteroskedasticity-consistent covariance estimators include McCaffrey and Bell (2002), Cribari-Neto (2004), MacKinnon (2011), Hausman and Palmer (2012) and Imbens and Kolesár (2016).

Recent studies have pointed out that the over-rejection problem remains problematic in considerably large samples when the data are leveraged/unbalanced. Based on simulation studies Young (2016) examines the distortion of the t-statistic due to interactions of the hypothesis and the data. Hansen (2017) derives the exact distribution of a t-type statistic, and through simulations show that using the "White Standard Errors" has significant size distortion when the data is highly leveraged. The basic idea behind both papers is that the nominal sample size may be "misleading" as the regression results may depend heavily on only a part of the observations rather than equally on all the data. Hence, the results are more variable and the test statistic tends to over-reject. Both Young (2016) and Hansen (2017) propose corrections

for t-type test statistic under i.i.d. normal disturbances. However, the proposed methods perform poorly when the disturbances are heteroskedastic. Furthermore, while the study of Young and others are based on simulations, Hansen derives the exact distribution of t-type statistic, which is in term of infinite series.

In view of these issues, this chapter compares the performance of the various heteroskedasticity-consistent variance estimators and proposes a F-type (t^2-type) test statistic for testing regression parameters under the heteroskedasticity of unknown form. This test statistic is valid for a single linear restriction on the regression parameter including the test for the zero restriction on each coefficient. It is shown that this proposed F-type test statistic can be expressed as a ratio of quadratic forms, and therefore its exact cumulative distribution under the null hypothesis can be easily written, and straightforwardly implemented from the result of Imhof (1961) on the distribution of quadratic form. For the applications of Imhof (1961) in a separate context of studying distributional properties of estimators see Bao et al. (2017), Ullah (2004), and Nakamura and Nakamura (1998), among others. A numerical calculation of the proposed test statistic, using Imhof (1961), is carried out to present the critical values and probability of rejections under various covariance estimators of regression estimators. The accuracies are confirmed from their corresponding simulation-based results. It is shown that the exact F-type test statistic based on Cribari-Neto (2004) estimator of covariance estimator provides the most accurate testing results, and it is followed by MacKinnon and White (1985b) covariance based test.

The remainder of this chapter is organized as follows. Section 2 studies the biases and variances of commonly used heteroskedasticity-consistent variance estimators under homoskedasticity. In Section 3, the heteroskedastic model and test statistic are presented. In Section 4, we derive the exact distribution for the test statistic under heteroskedasticity. In Section 5, we investigate the empirical size and power of the

test statistics using heteroskedasticity-consistent variance estimators by simulations. Section 6 concludes.

5.2 Performance of Heteroskedasticity-consistent Variance Estimators Under Homoskedasticity

In this section, we compare the performance of three heteroskedasticity-consistent variance estimators under homoskedasticity. First part is a analytical comparison of bias and variance of each estimator. The second part shows simulation results of the empirical performance.

5.2.1 Biases and Variances of Heteroskedasticity-consistent Variance Estimators

The setup of the model is as follows. The model is

$$\underset{n\times 1}{y} = \underset{n\times k}{X}\underset{k\times 1}{\beta} + \underset{n\times 1}{u}$$

where $u \sim N(0, I)$ are i.i.d. standard normal errors. The OLS estimator for β is

$$\hat{\beta} = (X'X)^{-1} X'y.$$

Hence, the variance of the OLS estimator is

$$\begin{aligned} V\left(\hat{\beta}|X\right) &= E\left((X'X)^{-1} X'uu'X (X'X)^{-1} |X\right) \\ &= (X'X)^{-1} X'\Omega X (X'X)^{-1}, \end{aligned}$$

where

$$\Omega = E\left(uu'\right).$$

The estimated residuals are

$$\hat{u} = y - X\hat{\beta} = (I - X\left(X'X\right)^{-1}X')y = My = Mu$$

where $M = I - X\left(X'X\right)^{-1}X'$ and

$$\hat{u}_i = m_i u$$

where m_i is the ith row of matrix M.

Next, we consider three commonly used estimators for $V\left(\hat{\beta}\right)$.

Hinkley (1977, HC_1)

In Hinkley (1977), the following estimator for $V\left(\hat{\beta}|X\right)$ is proposed

$$\hat{V}_{HC_1}\left(\hat{\beta}|X\right) = c\left(X'X\right)^{-1}X'\left\{\hat{u}_i^2\right\}X\left(X'X\right)^{-1}$$

where $c = \frac{n}{n-k}$ is a finite sample correction term and $\{\hat{u}_i^2\}$ is a diagonal matrix of $\hat{u}_i^2$. The expectation of the estimator $\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ is

$$\begin{aligned} E\left(\hat{V}_{HC_1}\left(\hat{\beta}\right)|X\right) &= E\left(c\left(X'X\right)^{-1}X'\left\{\hat{u}_i^2\right\}X\left(X'X\right)^{-1}|X\right) \\ &= c\left(X'X\right)^{-1}X'E\left(\left\{\hat{u}_i^2\right\}|X\right)X\left(X'X\right)^{-1} \end{aligned}$$

and

$$\begin{aligned} E\left(\hat{u}_i^2|X\right) &= E\left(m_i' u u' m_i|X\right) \\ &= m_i' E\left(uu'\right) m_i \\ &= m_i' \sigma^2 I m_i \\ &= \sigma^2 m_i' m_i \\ &= \sigma^2 m_{ii} \end{aligned}$$

where m_i denotes the ith column of $M = I - X\left(X'X\right)^{-1} X'$ and m_{ii} is the ith diagonal element of $M = I - X\left(X'X\right)^{-1} X'$. Note that $m_i' m_i = m_{ii}$ since M is symmetric and idempotent. Hence, under homoskedasticity, the expectation of $\hat{V}_{HC_1}\left(\hat{\beta}\right)$ is

$$\begin{aligned} E\left(\hat{V}_{HC_1}\left(\hat{\beta}\right)|X\right) &= c\left(X'X\right)^{-1} X' E\left(\left\{\hat{u}_i^2\right\}|X\right) X\left(X'X\right)^{-1} \\ &= c\left(X'X\right)^{-1} X' \left\{\sigma^2 m_{ii}\right\} X\left(X'X\right)^{-1} \\ &= \sigma^2 c\left(X'X\right)^{-1} X' \left\{m_{ii}\right\} X\left(X'X\right)^{-1} \end{aligned}$$

$\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ is unbiased if and only if $m_{ii} = \frac{n-k}{n}$ for all i (x is perfectly balanced). Otherwise, $\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ is upward or downward biased depending on x. The variance

of the estimator $\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ is positively related to

$$\begin{aligned} V\left(c\hat{u}_i^2|X\right) &= c^2 V\left(m_i' u u' m_i|X\right) \\ &= c^2 V\left(u' m_i m_i' u|X\right) \\ &= c^2\sigma^4\left[2tr\left(m_i m_i' m_i m_i'\right)\right] \\ &= c^2\sigma^4\left[2tr\left(m_i m_{ii} m_i'\right)\right] \\ &= c^2\sigma^4\left[2m_{ii} tr\left(m_i m_i'\right)\right] \\ &= c^2\sigma^4\left[2m_{ii}^2\right] \\ &= 2c^2\sigma^4 m_{ii}^2, \end{aligned}$$

which is equal to $2\sigma^4$ when x is perfectly balanced $\left(m_{ii} = \frac{n}{n-k}\right)$. We will compare the variance of $\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ with the variance of the other estimators later in the summary of this section.

MacKinnon and White (1985a, HC_2)

MacKinnon and White (1985a) propose two estimators for the variance of $\hat{\beta}$. The first one is

$$\hat{V}_{HC_2}\left(\hat{\beta}|X\right) = \left(X'X\right)^{-1} X'\left\{\frac{\hat{u}_i^2}{m_{ii}}\right\} X\left(X'X\right)^{-1}$$

where m_{ii} is the ith diagonal element of $M = I - X\left(X'X\right)^{-1}X'$, $\left\{\frac{\hat{u}_i^2}{m_{ii}}\right\}$ is a diagonal matrix of $\frac{\hat{u}_i^2}{m_{ii}}$. The expectation of the estimator $\hat{V}_{HC_2}\left(\hat{\beta}|X\right)$ is

$$\begin{aligned} E\left(\hat{V}_{HC_2}\left(\hat{\beta}\right)|X\right) &= E\left(\left(X'X\right)^{-1} X'\left\{\frac{\hat{u}_i^2}{m_{ii}}\right\} X\left(X'X\right)^{-1}|X\right) \\ &= \left(X'X\right)^{-1} X' E\left(\left\{\frac{\hat{u}_i^2}{m_{ii}}\right\}|X\right) X\left(X'X\right)^{-1} \end{aligned}$$

and

$$\begin{aligned} E\left(\frac{\hat{u}_i^2}{m_{ii}}|X\right) &= E\left(\frac{m_i' u u' m_i}{m_{ii}}|X\right) \\ &= m_i' E\left(\frac{uu'}{m_{ii}}|X\right) m_i \\ &= m_i' \frac{\sigma^2 I}{m_{ii}} m_i \\ &= \sigma^2 \frac{m_i' m_i}{m_{ii}} \\ &= \sigma^2 \end{aligned}$$

where m_i denotes the ith column of $M = I - X\left(X'X\right)^{-1} X'$ and m_{ii} is the ith diagonal element of $M = I - X\left(X'X\right)^{-1} X'$. Note that $m_i' m_i = m_{ii}$ since M is symmetric and idempotent. Hence, the expectation of the estimator

$$\begin{aligned} E\left(\hat{V}_{HC_2}\left(\hat{\beta}\right)|X\right) &= \left(X'X\right)^{-1} X' E\left(\left\{\frac{\hat{u}_i^2}{m_{ii}}\right\}|X\right) X\left(X'X\right)^{-1} \\ &= \left(X'X\right)^{-1} X' \left\{\sigma^2\right\} X\left(X'X\right)^{-1} \\ &= \left(X'X\right)^{-1} X' \sigma^2 I X\left(X'X\right)^{-1} \\ &= \sigma^2 \left(X'X\right)^{-1}. \end{aligned}$$

Hence, the estimator is unbiased. The variance of the estimator $\hat{V}_{HC_2}\left(\hat{\beta}|X\right)$ is positively related to

$$\begin{aligned}
V\left(\frac{\hat{u}_i^2}{m_{ii}}|X\right) &= V\left(\frac{m_i' u u' m_i}{m_{ii}}|X\right) \\
&= V\left(\frac{u' m_i m_i' u}{m_{ii}}|X\right) \\
&= \sigma^4\left[2tr\left(\frac{m_i m_i' m_i m_i'}{m_{ii}^2}\right)\right] \\
&= \sigma^4\left[2tr\left(\frac{m_i m_{ii} m_i'}{m_{ii}^2}\right)\right] \\
&= \sigma^4\left[2\frac{1}{m_{ii}} tr\left(m_i m_i'\right)\right] \\
&= \sigma^4\left[2\frac{1}{m_{ii}}\sum_j m_{ij}^2\right] \\
&= \sigma^4\left[2\frac{m_{ii}}{m_{ii}}\right] \\
&= 2\sigma^4.
\end{aligned}$$

We will compare the variance of $\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ with the variance of the other estimators later in the summary of this section.

MacKinnon and White (1985a, HC_3)

The second estimator for the variance of $\hat{\beta}$ proposed by MacKinnon and White (1985a) is

$$\hat{V}_{HC_3}\left(\hat{\beta}|X\right) = \left(X'X\right)^{-1} X' \left\{\frac{\hat{u}_i^2}{m_{ii}^2}\right\} X \left(X'X\right)^{-1}$$

where m_{ii} is the ith diagonal element of $M = I - X\left(X'X\right)^{-1}X'$, $\left\{\frac{\hat{u}_i^2}{m_{ii}^2}\right\}$ is a diagonal matrix of $\frac{\hat{u}_i^2}{m_{ii}^2}$. The expectation of the estimator is

$$\begin{aligned} E\left(\hat{V}_{HC_3}\left(\hat{\beta}\right)|X\right) &= E\left(\left(X'X\right)^{-1}X'\left\{\frac{\hat{u}_i^2}{m_{ii^2}}\right\}X\left(X'X\right)^{-1}|X\right) \\ &= \left(X'X\right)^{-1}X'E\left(\left\{\frac{\hat{u}_i^2}{m_{ii}^2}\right\}|X\right)X\left(X'X\right)^{-1} \end{aligned}$$

and

$$\begin{aligned} E\left(\frac{\hat{u}_i^2}{m_{ii}^2}|X\right) &= E\left(\frac{m_i'uu'm_i}{m_{ii}^2}|X\right) \\ &= m_i'E\left(\frac{uu'}{m_{ii}^2}|X\right)m_i \\ &= m_i'\frac{\sigma^2 I}{m_i^2}m_i \\ &= \sigma^2\frac{m_i'm_i}{m_{ii}^2} \\ &= \frac{\sigma^2}{m_{ii}}, \end{aligned}$$

where m_i denotes the ith column of $M = I - X\left(X'X\right)^{-1}X'$ and m_{ii} is the ith diagonal element of $M = I - X\left(X'X\right)^{-1}X'$. $m_i'm_i = m_{ii}$ since M is symmetric and idempotent. Hence, the expectation

$$\begin{aligned} E\left(\hat{V}_{HC_3}\left(\hat{\beta}\right)|X\right) &= \left(X'X\right)^{-1}X'E\left(\left\{\frac{\hat{u}_i^2}{m_{ii}^2}\right\}|X\right)X\left(X'X\right)^{-1} \\ &= \left(X'X\right)^{-1}X'\left\{\frac{\sigma^2}{m_{ii}}|X\right\}X\left(X'X\right)^{-1} \\ &= \sigma^2\left(X'X\right)^{-1}X'\left\{\frac{1}{m_{ii}}|X\right\}X\left(X'X\right)^{-1} \end{aligned}$$

Since M is symmetric and idempotent, we have that $m_{ii} = \sum_j m_{ij}^2 = m_{ii}^2 + \sum_{i\neq j} m_{jj}^2$, $m_{ii} \geq m_{ii}^2$, $0 \leq m_{ii} \leq 1$, $1 \leq \frac{1}{m_{ii}}$. Hence, the estimator $\hat{V}_{HC_3}\left(\hat{\beta}\right)$ is upward biased.

The variance of the estimator is positively related to

$$\begin{aligned}
V\left(\frac{\hat{u}_i^2}{m_{ii}^2}|X\right) &= V\left(\frac{m_i' u u' m_i}{m_{ii}^2}|X\right) \\
&= V\left(\frac{u' m_i m_i' u}{m_{ii}^2}|X\right) \\
&= \sigma^4\left[2tr\left(\frac{m_i m_i' m_i m_i'}{m_{ii}^4}\right)\right] \\
&= \sigma^4\left[2tr\left(\frac{m_i m_{ii} m_i'}{m_{ii}^4}\right)\right] \\
&= \sigma^4\left[2\frac{1}{m_{ii}^3} tr\left(m_i m_i'\right)\right] \\
&= \sigma^4\left[2\frac{1}{m_{ii}^3}\sum_j m_{ij}^2\right] \\
&= \sigma^4\left[2\frac{m_{ii}}{m_{ii}^3}\right] \\
&= \frac{2\sigma^4}{m_{ii}^2}.
\end{aligned}$$

We will compare the variance of $\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ with the variance of the other estimators later in the summary of this section.

Summary of biases and variances

In the previous sections, we analytically derived the biases and variances of the three heteroskedasticity-consistent variance estimators for the $\hat{\beta}$. The estimators are as follows.

$$\hat{V}_{HC_1}\left(\hat{\beta}|X\right) = c\left(X'X\right)^{-1} X'\left\{\hat{u}_i^2\right\} X\left(X'X\right)^{-1}$$

where $c = \frac{n}{n-k}$,

$$\hat{V}_{HC_2}\left(\hat{\beta}|X\right) = \left(X'X\right)^{-1} X'\left\{\frac{\hat{u}_i^2}{m_{ii}}\right\} X\left(X'X\right)^{-1},$$

and

$$\hat{V}_{HC_3}\left(\hat{\beta}|X\right) = (X'X)^{-1} X' \left\{ \frac{\hat{u}_i^2}{m_{ii}^2} \right\} X (X'X)^{-1}.$$

We analyze the biases and variances of the estimators under homoskedasticity. $\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ is unbiased if and only if $m_{ii} = \frac{n-k}{n}$ for all i. Otherwise, $\hat{V}_{HC_1}\left(\hat{\beta}|X\right)$ could be upward or downward biased. $\hat{V}_{HC_2}\left(\hat{\beta}|X\right)$ is unbiased for all m_{ii}. $\hat{V}_{HC_3}\left(\hat{\beta}|X\right)$ is always upward biased.

For the variances, since $m_{ii} = \sum_j m_{ij}^2 = m_{ii}^2 + \sum_{j \neq i} m_{ij}^2$, $m_{ii} \geq m_{ii}^2$, $0 \leq m_{ii} \leq 1$, $\frac{1}{m_{ii}} \geq 1$, we have the following relations $V\left(\frac{\hat{u}_i^2}{m_{ii}}|X\right) < V\left(\frac{\hat{u}_i^2}{m_{ii}^2}|X\right)$, $V\left(\hat{V}_{HC_2}\left(\hat{\beta}\right)|X\right) < V\left(\hat{V}_{HC_3}\left(\hat{\beta}\right)|X\right)$. $V\left(c\hat{u}_i^2|X\right) = 2c^2\sigma^4 m_{ii}^2$ depends on the actual structure of the data. If $m_{ii} = \frac{n-k}{n}$, $V\left(\hat{V}_{HC_1}\left(\hat{\beta}|X\right)\right)$ is the same as $V\left(\hat{V}_{HC_2}\left(\hat{\beta}|X\right)\right)$.

In summary, $\hat{V}_{HC_2}\left(\hat{\beta}|X\right)$ is unbiased and has a smaller variance than $\hat{V}_{HC_3}\left(\hat{\beta}|X\right)$, thus $\hat{V}_{HC_2}\left(\hat{\beta}\right)$ is strictly better than $\hat{V}_3\left(\hat{\beta}\right)$ under homoskedasticity. The performance of $\hat{V}_{HC_1}\left(\hat{\beta}\right)$ depends on the structure of the actual data. If the data is perfectly balanced, $\hat{V}_{HC_1}\left(\hat{\beta}\right)$ performs exactly the same as $\hat{V}_2\left(\hat{\beta}\right)$. Hence, the performance of the estimators depends on the data, and it is impossible to draw the conclusion that any estimator is strictly better than the other two.

Simulation studies

In this section, we compare the performance of the three heteroskedasticity-consistent variance estimators under homoskedasticity. The first model is a small sample setup. The model is $y = \underset{100\times 10}{X}\beta + u$ where $u \sim N(0, I)$ and X are drawn from $N(0, 10)$. Simulation has 100 sets of X and 100 sets of u for each set of X.

Estimator	Bias	Variance	MSE
$c_{hc_1}(X'X)^{-1}X'\{\hat{u}_i^2\}X(X'X)^{-1}$	3.0323e-06	7.4030e-09	7.4695e-09
$(X'X)^{-1}X'\left\{\frac{\hat{u}_i^2}{m_{ii}}\right\}X(X'X)$	2.3940e-06	7.9606e-09	7.9716e-09
$(X'X)^{-1}X'\left\{\frac{\hat{u}_i^2}{m_{ii}^2}\right\}X(X'X)^{-1}$	1.5182e-05	1.0677e-08	1.3061e-08
$\frac{\sum\hat{u}_i^2}{n}(X'X)^{-1}$	1.1409e-05	2.3220e-09	3.6641e-09

The second model is a large sample setup. The model is $y = \underset{1000\times1}{X}\beta + u$ where $u \sim N(0, I)$ and X are drawn from $N(0, 10)$. Simulation has 100 sets of X and 100 sets of u for each X.

Estimator	Bias	Variance	MSE
$c_{hc_1}(X'X)^{-1}X'\{\hat{u}_i^2\}X(X'X)^{-1}$	5.8946e-08	3.4458e-12	3.4493e-12
$(X'X)^{-1}X'\left\{\frac{\hat{u}_i^2}{m_{ii}}\right\}X(X'X)$	2.4430e-08	3.4674e-12	3.4680e-12
$(X'X)^{-1}X'\left\{\frac{\hat{u}_i^2}{m_{ii}^2}\right\}X(X'X)^{-1}$	7.8176e-08	3.4823e-12	3.4884e-12
$\frac{\sum\hat{u}_i^2}{n}(X'X)^{-1}$	2.1942e-08	1.8388e-12	1.8392e-12

Simulation results show that the default OLS estimator has the smallest bias as well as variance. This is expected since the true error are homoskedastic. Among the three heteroskedastic consistent estimators, the first estimator which has a finite sample correction term is shown to has the smallest mean square error. According to our analytical result, this should partly result from the well-balanced structure of our simulated data. In practice, we may expect unbalanced data which would lead to a larger bias as well as variance for the Hinkley (1977) estimator. The second estimator which has been proved to be unbiased has the smallest bias. The third estimator performs worst in small sample. However, when sample size gets larger, the estimators perform almost equally.

5.2.2 Distribution of t-statistics using Heteroskedasticity-consistent Variance Estimators

In this section, we investigate the performance of t-statistics using the three heteroskedasticity-consistent variance estimators. We use the same setup as the previous section. The t-statistics for $H_0 : r'(\beta - \beta_0) = 0$ is calculated as below.

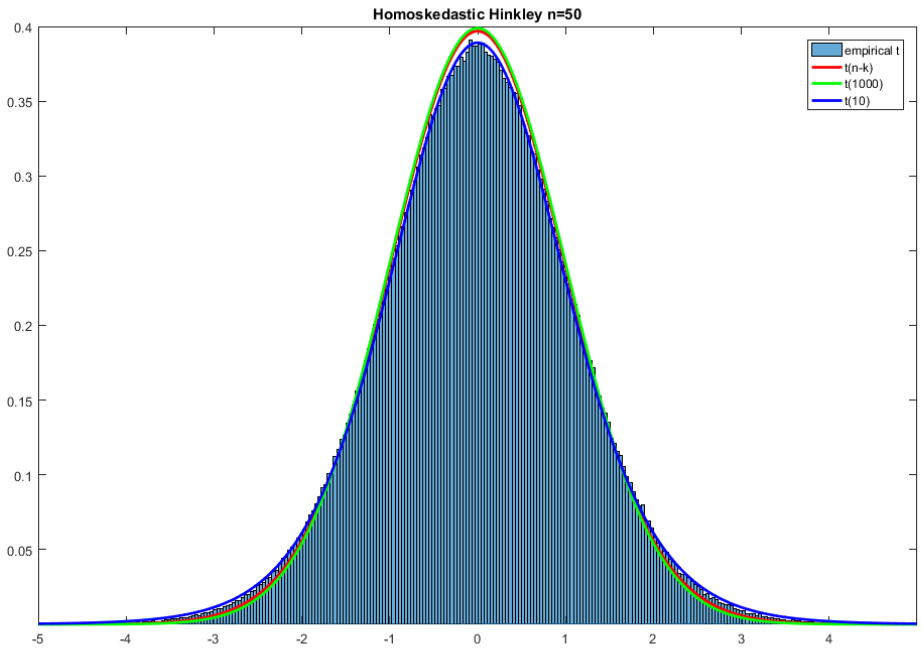
$$t = \frac{r'(\hat{\beta} - \beta_0)}{\sqrt{r'\hat{\Sigma}r}}$$

where $\hat{\Sigma}$ is the variance estimator to be investigated.

The distributions of the t-statistic are shown below. In the graphs, the histogram is the empirical distribution of the above t-statistics calculated using the respective variance-covariance estimator. The solid red line is the probability density function of t-distribution with $n-k$ degrees of freedom. The solid green line is the probability density function of t-distribution with 1000 degrees of freedom. The solid blue line is the probability density function of t-distribution with 10 degrees of freedom.

For the t-statistics using the Hinkley (1977) variance-covariance estimators, we can see that in the center, the red and the green lines is obviously above the histogram, however, the blue line fits the boundary of the histogram pretty well. In the tails, the red and the green lines are below the histogram and the blue line again fits the boundary of the histogram very well. A numerical comparison of the integrated difference between the PDFs and the empirical distribution is shown at the end of this section. The results show that the distribution of the t-statistics using the Hinkley (1977) variance-covariance estimator is likely to be a t-distribution with fewer degrees of freedom than suggested by theory. In our case, the suggested degrees of freedom $n-k = 50-3 = 47$ is much larger than the actual degrees of freedom of the empirical

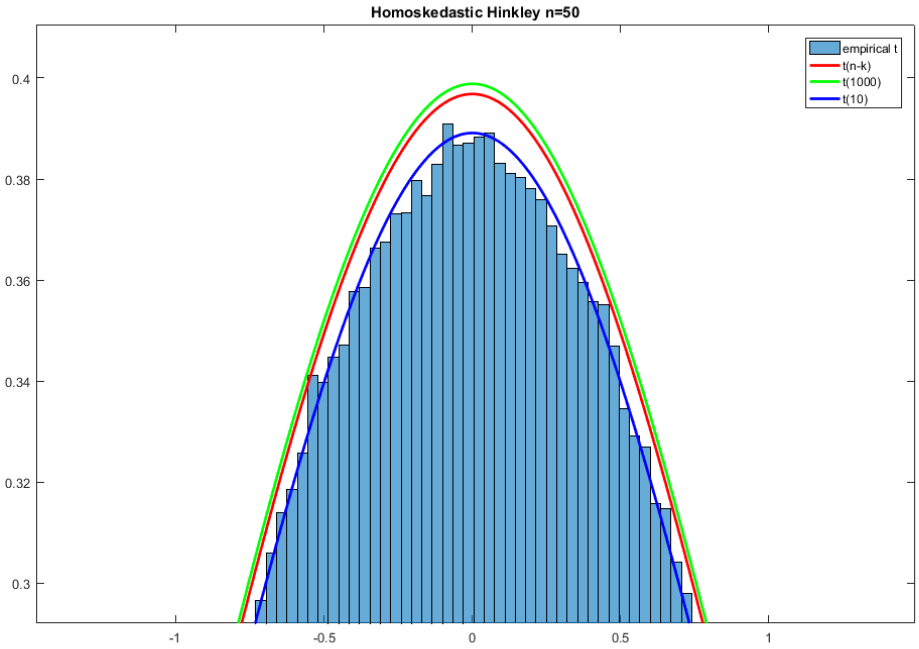
Figure 5.1: t-statistics using Hinkley (1977, HC_1)



distribution.

For the t-statistics using the first MacKinnon and White (1985a) variance-covariance estimators, we can see that in the center, the green line is slightly above the histogram, however, the blue line is completely contained in the histogram. The red line in this case fits the boundary of the histogram almost perfectly. In the tails, the behavior for the green and blue lines are reversed. The green line is below the histogram and the blue line is above the histogram. Again, the red line fits the boundary of the histogram the best. The results show that the distribution of the t-statistics using the first MacKinnon and White (1985a) variance-covariance estimator is a t-distribution with exactly the same degrees of freedom as suggested by theory.

Figure 5.2: Tails and Center Behavior



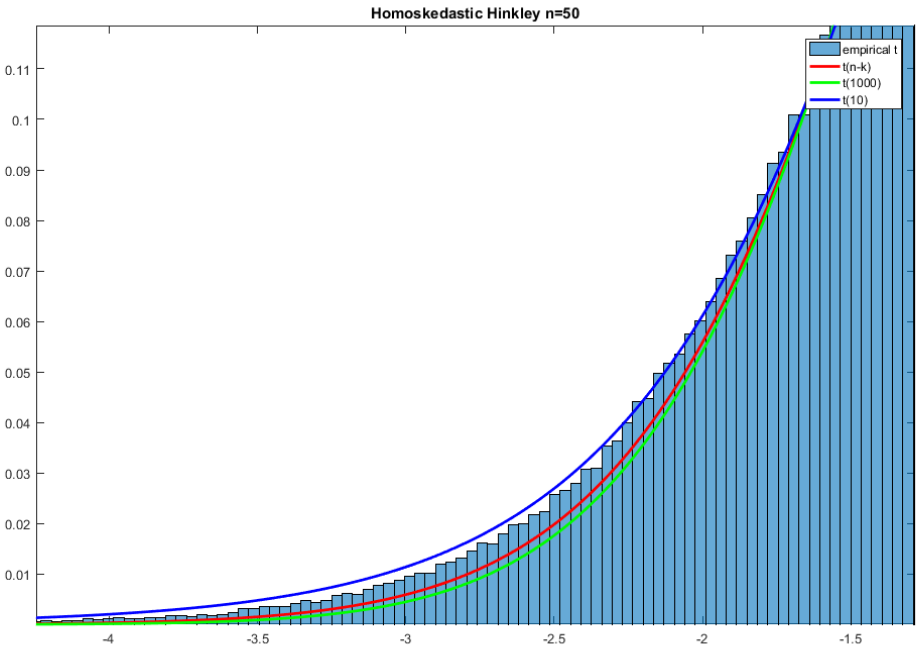
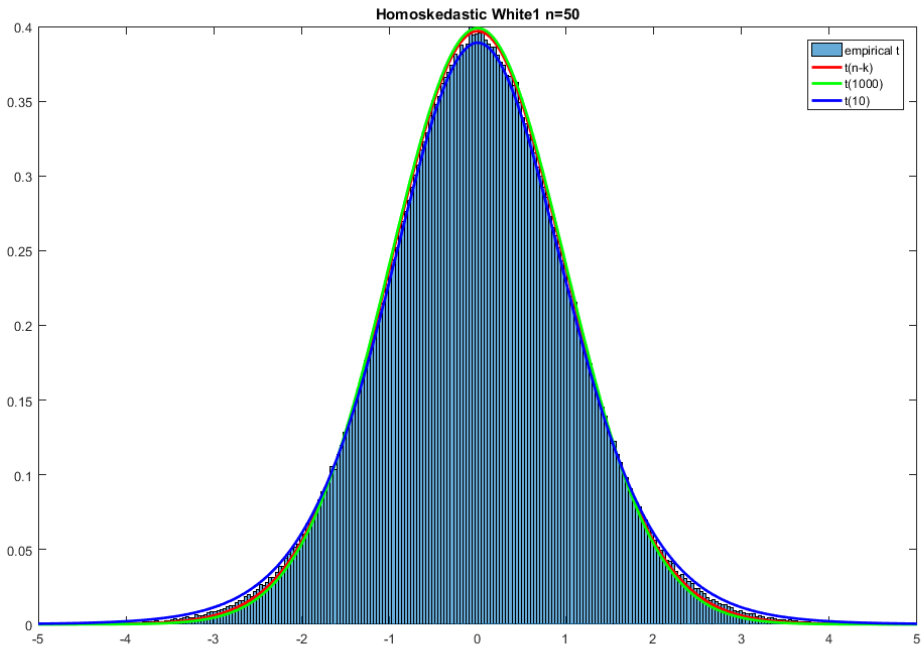


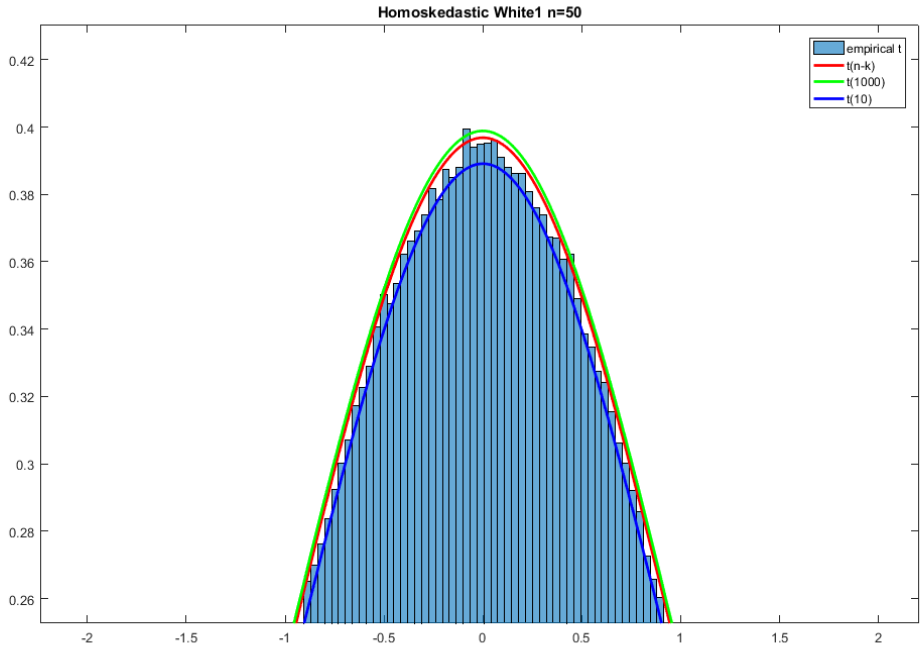
Figure 5.3: t-statistics using MacKinnon and White (1985a, HC_2)



For the t-statistics using the second MacKinnon and White (1985a) variance-covariance estimators, we can see that in the center, all the solid lines are significantly below the histogram. From the graph, it's obvious that there is no t-distribution will fit the empirical distribution since the height of the center is much higher than the normal distribution. In the tails, the blue line is significantly above the histogram. The green line, on the other hand, is slightly below the histogram. The red line seems to be a pretty good fit of the histogram in the tails. The results show that the distribution of the t-statistics using the second MacKinnon and White (1985a) variance-covariance estimator is not t-distribution at all.

In the tables below, we provide a numerical comparison of the difference between the empirical distributions and the three proposed theoretical distributions where the

Figure 5.4: Tails and Center Behavior



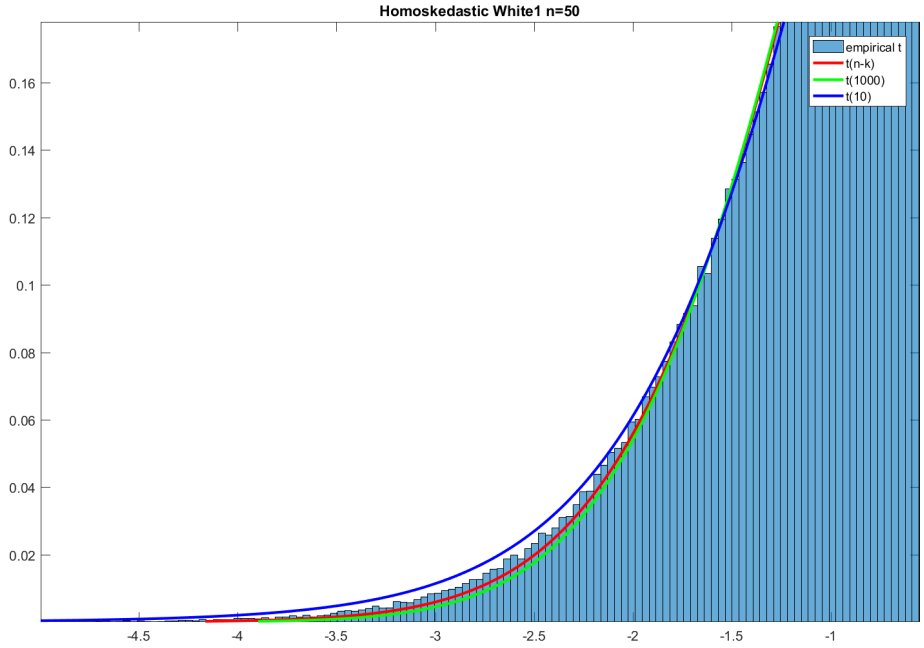
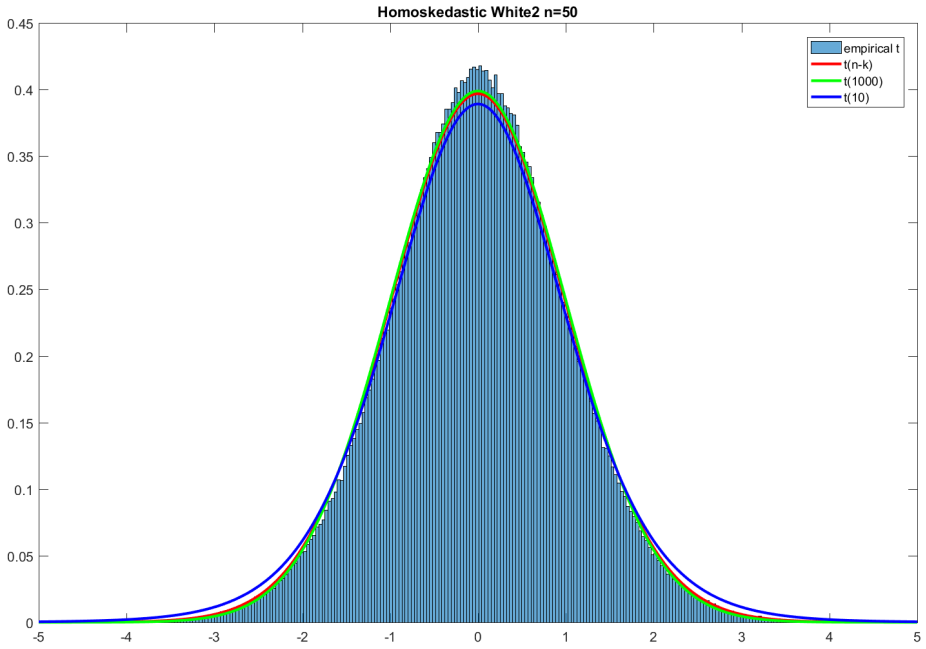


Figure 5.5: t-statistics using MacKinnon and White (1985a, HC_3)



number shown in the table is the integrated difference

$$\int \left| \hat{f}(x) - f(x) \right| dx.$$

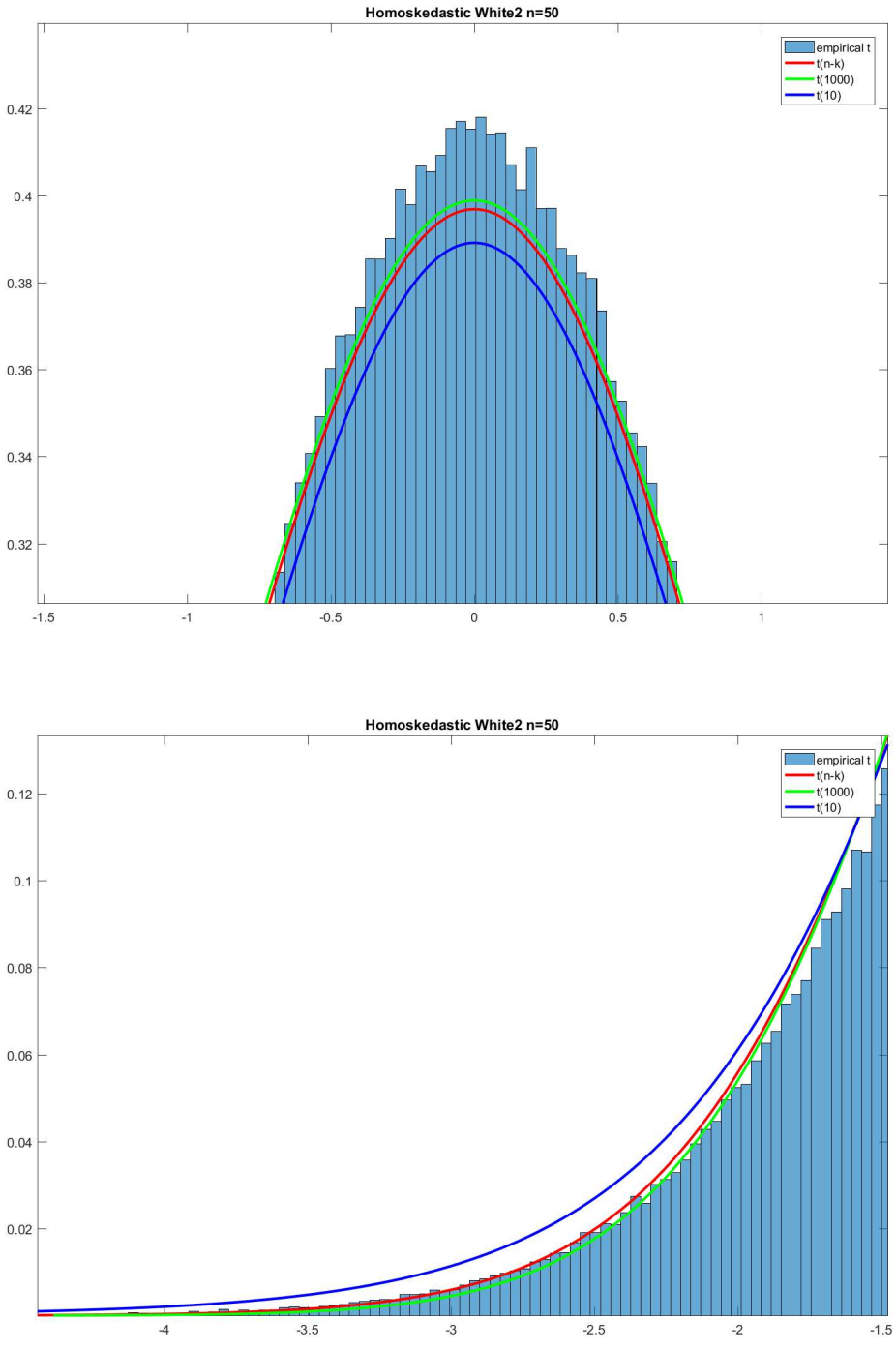
Table 5.1: $n = 50$

degree of freedom	10	$n-k$	1000
Hinkley (HC_1)	0.0062	0.0067	0.0074
MacKinnon and White (HC_2)	0.0064	0.0062	0.0067
MacKinnon and White (HC_3)	0.0086	0.0066	0.0066

Table 5.2: $n = 100$

degree of freedom	10	$n-k$	1000
Hinkley (HC_1)	0.0070	0.0061	0.0063
MacKinnon and White (HC_2)	0.0072	0.0059	0.0060
MacKinnon and White (HC_3)	0.0083	0.0059	0.0059

Figure 5.6: Tails and Center Behavior



5.3 Model and Test Statistic under Heteroskedasticity

As shown in the previous sections, the distribution of the test statistic is ill-behaved when heteroskedasticity-consistent variance estimators are used even in the homoskedastic case. Hence, the results of statistical inferences may be misleading. Next, we propose a new method for statistical inference under heteroskedasticity. For the model

$$y_i = \beta_0 + \sum_{k=1}^{p} \beta_k \mathbf{X}_{ik} + u_i, \quad u_i = \sigma_i \epsilon_i, \quad \epsilon_i \sim N(0,1), \tag{5.1}$$

with $\text{Var}(u) \equiv \mathbf{\Omega} = \text{diag}\{\sigma_i^2\}$, the variance of the parameter vector $\hat{\beta}$ estimated by OLS is

$$\text{Var}(\hat{\beta}) := \mathbf{\Sigma} = (\mathbf{X'X})^{-1}\mathbf{X'\Omega X}(\mathbf{X'X})^{-1}. \tag{5.2}$$

We consider the test statistics of the form

$$F = t^2 = \left(\frac{\mathbf{r'}(\hat{\beta} - \beta_\mathbf{0})}{\sqrt{\mathbf{r'\hat{\Sigma}r}}} \right)^2 = \frac{\mathbf{u'X}(\mathbf{X'X})^{-1}\mathbf{rr'}(\mathbf{X'X})^{-1}\mathbf{X'u}}{\mathbf{r'}(\mathbf{X'X})^{-1}\mathbf{X'\hat{\Omega}X}(\mathbf{X'X})^{-1}\mathbf{r}} \tag{5.3}$$

corresponding to a null hypothesis about a linear combination of the estimated parameters $H_0 : \mathbf{r'}\beta = \mathbf{r'}\beta_\mathbf{0} = a$, where $\hat{\mathbf{\Sigma}}$ is an asymptotically consistent estimate of $\text{Var}(\hat{\beta})$. The following approaches to estimate $\mathbf{X'\Omega X}$ are proposed in the literature.

- HC_0 is the original formulation used in White standard errors. White (1980) recognizes that $\mathbf{X'\hat{\Omega}X}$ is a consistent estimator of $\mathbf{X'\Omega X}$ when using the sample matrix

  $$\hat{\mathbf{\Omega}}_0 = \text{diag}\left\{\hat{u}_i^2\right\}.$$

  where $\hat{u}_i^2$ are the fitted residuals from estimating (5.1) via OLS.

- HC_1 (Hinkley, 1977) adjusts for degrees of freedom and is the most commonly

used robust standard error estimator

$$\hat{\mathbf{\Omega}}_1 = \frac{n}{n-p}\text{diag}\left\{\hat{u}_i^2\right\}.$$

- HC_2 (MacKinnon and White, 1985b) adjusts for the leverage values h_i where h is the diagonal of the projection matrix $\mathbf{P_X} = \mathbf{X}(\mathbf{X'X})^{-\mathbf{1}}\mathbf{X'}$. It is an unbiased estimator of $\mathbf{X'\Omega X}$ under homoskedasticity.

$$\hat{\mathbf{\Omega}}_2 = \text{diag}\left\{\frac{\hat{u}_i^2}{1-h_i}\right\}.$$

- HC_3 (Davidson and MacKinnon, 1993) is an approximation to HC_j and is a slight modification of HC_2

$$\hat{\mathbf{\Omega}}_3 = \text{diag}\left\{\left(\frac{\hat{u}_i}{1-h_i}\right)^2\right\}.$$

- HC_4 (Cribari-Neto, 2004) adjusts the residuals by a leverage factor that increases with the leverage.

$$\hat{\mathbf{\Omega}}_4 = \text{diag}\left\{\frac{\hat{u}_i^2}{(1-h_i)^{\delta_i}}\right\}$$

where $\delta_i = \min\{4, \frac{nh_i}{p}\}$.

We consider these approaches in terms of their size in Section 5.5 below. MacKinnon (2011) show that each of the HC estimators continues to have significant size distortions even when n is of moderate size.

5.4 Exact Distribution of F-statistic using HC Estimators

In this section, we find the exact distribution of the above mentioned F-type test statistic by the numerical method proposed in Imhof (1961). Consider a quadratic form

$$Q = \mathbf{u}'\mathbf{N}\mathbf{u} = \sum_{r=1}^{m} \lambda_r \chi^2_{h_r} \tag{5.4}$$

where $\mathbf{u} \sim N(\mathbf{0}, \mathbf{\Omega})$, λ_r's are the distinct non-zero characteristic roots of $\mathbf{N\Omega}$, the h_r's their respective orders of multiplicity, and the $\chi^2_{h_r}$ are independent χ^2-variables with h_r degrees of freedom.

Imhof (1961) shows that the cumulative distribution of the quadratic form Q defined in (5.4) can be obtained quite easily by straightforward numerical integration of an inversion formula

$$\Pr[Q > c] = \frac{1}{2} + \frac{1}{\pi} \int_0^{\infty} \frac{\sin \theta(u)}{u\rho(u)} du, \tag{5.5}$$

where

$$\theta(u) = \frac{1}{2} \sum_{r=1}^{m} \left[h_r \tan^{-1}(\lambda_r u) + \delta_r^2 \lambda_r u (1 + \lambda_r^2 u^2)^{-1} \right] - \frac{1}{2} cu, \tag{5.6}$$

$$\rho(u) = \prod_{r=1}^{m} \left(1 + \lambda_r^2 u^2\right)^{\frac{1}{4} h_r} \exp\left\{ \frac{1}{2} \sum_{r=1}^{m} \frac{(\delta_r \lambda_r u)^2}{1 + \lambda_r^2 u^2} \right\}. \tag{5.7}$$

For each of the HC estimators mentioned in Section 5.3, we write the test statistic in (5.3) as a quadratic form of the disturbances $\mathbf{u}$ in (5.1). Let

$$\hat{\mathbf{\Omega}}_{\mathbf{j}} = \text{diag}\left\{ w_{ji} \hat{u}_i^2 \right\} \tag{5.8}$$

where $j \in \{1, 2, 3, 4\}$ is an index for the type of estimator HC_j and

$$
\begin{aligned}
w_{0i} &= 1, && (5.9)\\
w_{1i} &= \frac{n}{n-p}, && (5.10)\\
w_{2i} &= \frac{1}{1-h_i}, && (5.11)\\
w_{3i} &= \frac{1}{(1-h_i)^2}, && (5.12)\\
w_{4i} &= \frac{1}{(1-h_i)^{\delta_i}}, && (5.13)
\end{aligned}
$$

where $\delta_i = \min\{4, \frac{nh_i}{p}\}$. Hence, the test statistic (5.3) using HC_j

$$
\begin{aligned}
F_j &= \frac{\mathbf{u'X(X'X)^{-1}rr'(X'X)^{-1}X'u}}{\mathbf{r'(X'X)^{-1}X'\hat{\Omega}}_j\mathbf{X(X'X)^{-1}r}} && (5.14)\\
&= \frac{\mathbf{u'X(X'X)^{-1}rr'(X'X)^{-1}X'u}}{\mathbf{r'(X'X)^{-1}X'}\,\mathrm{diag}\{w_{ji}\hat{u}_i^2\}\mathbf{X(X'X)^{-1}r}} && (5.15)\\
&= \frac{\mathbf{u'zz'u}}{\sum_{i=1}^n w_{ji}z_i^2\hat{u}_i^2} && (5.16)\\
&= \frac{\mathbf{u'zz'u}}{\mathbf{\hat{u}'}\,\mathrm{diag}\{w_{ji}z_i^2\}\mathbf{\hat{u}}} && (5.17)\\
&= \frac{\mathbf{u'zz'u}}{\mathbf{u'M}\,\mathrm{diag}\{w_{ji}z_i^2\}\mathbf{Mu}} && (5.18)\\
&= \frac{\mathbf{u'Au}}{\mathbf{u'B}_j\mathbf{u}} && (5.19)
\end{aligned}
$$

where $\mathbf{z} = \mathbf{X(X'X)^{-1}r}$, $\mathbf{M} = \mathbf{I_n} - \mathbf{X(X'X)^{-1}X'}$, $\mathbf{A} = \mathbf{zz'}$ and $\mathbf{B}_j = \mathbf{M}\,\mathrm{diag}\{w_{ij}z_i^2\}\mathbf{M}$. Hence,

$$
\begin{aligned}
\Pr[F_j > c|\mathbf{X}] &= \Pr\left[\frac{\mathbf{u'Au}}{\mathbf{u'B}_j\mathbf{u}} > c \middle| \mathbf{X}\right] && (5.20)\\
&= \Pr\left[\mathbf{u'}(\mathbf{A} - c\mathbf{B}_j)\mathbf{u} > 0|\mathbf{X}\right]. && (5.21)\\
&= \Pr\left[\mathbf{u'N}_{c,j}\mathbf{u} > 0|\mathbf{X}\right] && (5.22)
\end{aligned}
$$

where $\mathbf{N}_{c,j} = \mathbf{A} - c\mathbf{B}_j$ in equation (5.4), the cumulative distribution of the test statistic (5.3) can be approximated by approximating (5.21) using (5.5). The critical value of the test statistic at the α significance level, hence, can be obtain by

$$c_{j,\alpha} = \text{Imhof}(\alpha, \mathbf{N}_j\boldsymbol{\Omega}) := \arg_c\{\Pr\left[\mathbf{u}'\mathbf{N}_j\mathbf{u} > 0|\mathbf{X}\right] = \alpha\}. \tag{5.23}$$

Table 5.3 shows the rejection rate of the test statistics F_j when the critical values are obtained by (5.23). The rejection rates when the critical values are obtained from (5.23) are identical (up to some approximation errors) to the nominal rate α. This is not surprising once we take into account the fact that, in the approximation, we have used $\boldsymbol{\Omega}$ which is the true covariance matrix of the disturbances $\mathbf{u}$. And if $\boldsymbol{\Omega}$ is known, the test statistic in (5.3) using $\boldsymbol{\Omega}$

$$F^* = \frac{\mathbf{u}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{r}\mathbf{r}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{u}}{\mathbf{r}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Omega}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{r}} = \left(\frac{\mathbf{r}'(\hat{\beta} - \beta_\mathbf{0})}{\sqrt{\mathbf{r}'\boldsymbol{\Sigma}\mathbf{r}}}\right)^2 \sim \chi^2_1 \tag{5.24}$$

is known to follow the χ^2-distribution with 1 degrees of freedom or, equivalently, the limiting distribution of the F-distribution with 1 and $n - p$ degrees of freedom when $n \to \infty$.

5.5 Empirical Size and Power

In this section, we present the empirical size and power of the tests using HC estimators through extensive simulation studies. Many papers that use simulation to study the properties of HC estimators, beginning with MacKinnon and White (1985b) and extending at least to Cribari-Neto and Lima (2010), have simply chosen a fixed or random $\mathbf{X}$ matrix for a small sample size – just 10 in the case of Davidson

Table 5.3: Rejection Rate using Imhof (1961) with True $\mathbf{\Omega}$

n		100			50	
α	0.01	0.05	0.10	0.01	0.05	0.10
F_{0F}	0.00855	0.04574	0.10508	0.01411	0.06571	0.13093
F_0	0.01015	0.05032	0.09991	0.00967	0.04899	0.09941
F_1	0.01015	0.05032	0.09991	0.00967	0.04899	0.10049
F_2	0.01002	0.05036	0.09999	0.00954	0.04876	0.10051
F_3	0.00998	0.05031	0.10012	0.00958	0.04860	0.09950
F_4	0.01005	0.05030	0.09984	0.00942	0.04785	0.09875

F_{0F} is identical to F_0 except critical values are obtained from the F-table.
The details of the simulation experiment are given in Section 5.5 DGP1.

and Flachaire (2008) – and formed larger samples by repeating it as many times as necessary. When $\mathbf{X}$ matrices are generated in this way, there will only be as many distinct values of h_i as the number of observations in the original sample. Moreover, all of those values, and in particular the largest one, must be exactly proportional to $\frac{1}{n}$; see Chesher (1989). This ensures that inference based on heteroskedasticity-robust methods improves very rapidly as n increases. Since very few real datasets involve $\mathbf{X}$ matrices for which all of the h_i are proportional to $\frac{1}{n}$, this sort of experiment almost certainly paints an excessively optimistic picture.

In contrast, MacKinnon (2011) draws the regressors from a lognormal distribution so as to make heteroskedasticity-robust inference difficult. Because many samples will contain a few observations on the $\mathbf{X}$ that are quite extreme, and the most extreme observation in each sample will tend to become more so as the sample size increases. Therefore, the largest value of h_i will tend to be large and to decline very slowly as $n \to \infty$. In fact, the average value of h_i^{max} is nearly 0.80 when $n = 20$ and declines by a factor of only about 3.5 as the sample size increase to 1280, with the rate of decline increasing somewhat as n becomes larger. It is likely that few real datasets have h_i which are as badly behaved as the ones in these experiments, so their results

certainly paint an excessively pessimistic picture.

We consider a simple linear model similar to the survey paper of MacKinnon (2011)

$$y_i = \beta_0 + \beta_1 x_i + u_i, \quad u_i = \sigma_i \epsilon_i, \quad \epsilon_i \sim N(0,1), \tag{5.25}$$

and

$$\text{DGP: } \sigma_i^2 \;=\; \exp(x_i^2), \tag{5.26}$$

$$\tag{5.27}$$

where $\beta = (0,0)$ and $x_i \sim N(0,1)$. The $\mathbf{X}$ matrix is drawn from the normal distribution in order to imitate real datasets with the greatest generality.

The results are reported in Tables 5.4, 5.5, 5.6, 5.7, and 5.8. Table 5.4 shows the empirical size of the F-statistic using heteroskedasticity-consistent variance estimators. The rows of F_i shows the empirical size of the test using critical values from the F-table. The rows of HC_i show the empirical sizes of the test using the corrected critical values obtained through our proposed method. We can see that the test using our corrected critical values has much better size than the standard method of using F-table. Table 5.5, 5.6, 5.7, and 5.8 show the power of the test using our corrected critical values compared with using the critical values from the F-table. We see that around the null, the test using critical values from the F-table has higher power which is consistent to the fact that using the critical values from the F-table tends to significantly over-reject under the null. However, as we deviate away from the null, we see that the power of the test using the corrected critical values immediately catches up with the power of the test using the critical values from the F-table. Hence, we draw the conclusion that our test using the corrected critical values have significantly

better size while does not lose too much power. Hence, the corrected critical values should be an ideal substitute for the standard method.

5.6 Conclusions

In this chapter, the performance of different heteroskedasticity-consistent variance estimators are studied both under homoskedasticity and heteroskedasticity. We provide the analytical bias and variance of different heteroskedasticity-consistent variance estimators under homoskedasticity as well as numerical analysis. Under heteroskedasticity, we use the Imhof (1961) technique to approximate the distribution of the F-type test statistic using heteroskedasticity-consistent variance estimators and propose corrected methods for getting the correct critical values of the test statistic. Monte Carlo results show that the proposed method has better size and power than standard methods.

Table 5.4: Size of Test

n	15				30				50				100			
α	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00
F0:	0.06438	0.17516	0.27647	0.02766	0.09466	0.16532	0.0148	0.07234	0.1339	0.00851	0.04998	0.10378				
HC0:	0.03918	0.08417	0.14926	0.01762	0.05783	0.10049	0.01062	0.05099	0.09498	0.00827	0.04642	0.09146				
F1:	0.04773	0.14381	0.23752	0.02328	0.08302	0.147	0.01286	0.06616	0.12499	0.00785	0.04698	0.10004				
HC1:	0.03866	0.08417	0.14925	0.01762	0.05783	0.10049	0.01062	0.05099	0.09498	0.00827	0.04642	0.09146				
F2:	0.03361	0.09256	0.18251	0.01765	0.06592	0.11802	0.01055	0.05439	0.10435	0.00669	0.04164	0.08968				
HC2:	0.03329	0.07414	0.12766	0.01625	0.05441	0.0958	0.0101	0.04918	0.09204	0.00802	0.04574	0.09032				
F3:	0.01825	0.05482	0.10628	0.0111	0.04546	0.08636	0.0077	0.04108	0.08058	0.00538	0.03493	0.07778				
HC3:	0.02958	0.06676	0.11944	0.01503	0.05186	0.09249	0.00969	0.0475	0.08979	0.00782	0.04504	0.08933				
F4:	0.01637	0.04602	0.09364	0.00901	0.03675	0.07201	0.0061	0.03131	0.05913	0.00413	0.02724	0.06361				
HC4:	0.02637	0.05924	0.114	0.01316	0.04781	0.08887	0.00897	0.04444	0.08592	0.00729	0.04372	0.08744				
Homo:	0.01827	0.07339	0.16444	0.01149	0.0562	0.11217	0.00814	0.04961	0.10223	0.00579	0.04011	0.08982				

All F's are the rejection rate based on critical values given in the F-table.

Table 5.5: Power of Test

$n = 15$ $k = 2$ Replication= 1000×100

α	0.01				0.05				0.10			
a	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00
F0:	0.05907	0.18418	0.81741	0.90373	0.17304	0.36016	0.85648	0.93635	0.28819	0.46923	0.90887	0.95366
HC0:	0.03490	0.13120	0.73195	0.85652	0.09113	0.22487	0.76953	0.89399	0.13759	0.30044	0.84778	0.90328
F1:	0.04324	0.15557	0.79979	0.89131	0.14076	0.31731	0.84425	0.93210	0.24167	0.42613	0.90239	0.95021
HC1:	0.03460	0.13016	0.72532	0.85096	0.09098	0.22476	0.7695	0.89295	0.13759	0.30044	0.84778	0.90328
F2:	0.02899	0.12707	0.76122	0.87047	0.09975	0.24925	0.81495	0.91777	0.17155	0.35021	0.8837	0.92584
HC2:	0.03052	0.11733	0.70163	0.83166	0.08175	0.21104	0.7532	0.87956	0.12622	0.28147	0.8364	0.89724
F3:	0.01636	0.08449	0.69782	0.83543	0.06057	0.18352	0.75596	0.88382	0.10570	0.26243	0.84115	0.90054
HC3:	0.02742	0.10672	0.68197	0.81458	0.07522	0.20082	0.73959	0.87047	0.11828	0.26911	0.82942	0.89407
F4:	0.01570	0.07821	0.65686	0.79016	0.05474	0.17457	0.69485	0.83893	0.09247	0.23678	0.79931	0.86548
HC4:	0.02431	0.09432	0.65510	0.79216	0.06880	0.19063	0.72584	0.86763	0.11272	0.26215	0.82978	0.89605
Homo:	0.01671	0.09010	0.73418	0.85808	0.08057	0.21870	0.80364	0.91523	0.15234	0.33123	0.88031	0.92675

Table 5.6: Power of Test

$n = 30$	$k = 2$	Replication= 1000×100											
α		0.01				0.05				0.10			
a	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00
F0:	0.02648	0.15583	0.77667	0.91269	0.09945	0.30295	0.85188	0.94898	0.18663	0.38517	0.88116	0.95418
HC0:	0.01708	0.12183	0.74241	0.89465	0.06040	0.23734	0.82263	0.93443	0.11526	0.30138	0.84183	0.93807
F1:	0.02209	0.14252	0.76845	0.90923	0.08759	0.28476	0.84642	0.94665	0.16874	0.36638	0.87550	0.95261
HC1:	0.01708	0.12183	0.74241	0.89465	0.06040	0.23734	0.82263	0.93443	0.11526	0.30138	0.84183	0.93807
F2:	0.01672	0.12435	0.74794	0.89757	0.06913	0.25417	0.83188	0.93755	0.13698	0.32791	0.85560	0.94455
HC2:	0.01551	0.11624	0.73526	0.89004	0.05671	0.23031	0.81885	0.93185	0.10958	0.29404	0.83840	0.93674
F3:	0.01085	0.09845	0.71851	0.87980	0.04810	0.21333	0.80905	0.92497	0.10001	0.27888	0.83069	0.93361
HC3:	0.01451	0.11119	0.72834	0.88573	0.05402	0.22433	0.81538	0.92987	0.10498	0.28807	0.83621	0.93576
F4:	0.00852	0.08759	0.68139	0.85056	0.03834	0.18831	0.77149	0.89764	0.07477	0.23873	0.79301	0.91032
HC4:	0.01294	0.10338	0.71163	0.87461	0.04970	0.21463	0.80870	0.92712	0.09924	0.28095	0.83455	0.93517
Homo:	0.01112	0.10207	0.73681	0.89237	0.05967	0.23931	0.82851	0.93772	0.12938	0.32130	0.85755	0.94591

Table 5.7: Power of Test

$n = 50$	$k = 2$	Replication $= 1000 \times 100$											
α		0.01				0.05				0.10			
a	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00
F0:	0.01731	0.14228	0.74285	0.85856	0.07503	0.28098	0.85444	0.91898	0.14473	0.37649	0.83665	0.91122
HC0:	0.01189	0.12429	0.73668	0.85322	0.05584	0.24750	0.84598	0.91515	0.10567	0.33170	0.81820	0.90101
F1:	0.01511	0.13433	0.73721	0.85581	0.06870	0.27066	0.85087	0.91702	0.13627	0.36612	0.83286	0.90937
HC1:	0.01189	0.12429	0.73668	0.85322	0.05584	0.24750	0.84598	0.91515	0.10567	0.33170	0.81820	0.90101
F2:	0.01211	0.12143	0.72130	0.84529	0.05853	0.25083	0.83982	0.90997	0.11662	0.34308	0.81914	0.90127
HC2:	0.01125	0.12107	0.73202	0.85031	0.05397	0.24419	0.84416	0.91412	0.10290	0.32874	0.81685	0.90026
F3:	0.00829	0.10268	0.69813	0.83123	0.04535	0.22247	0.82385	0.90076	0.09259	0.31136	0.80087	0.89122
HC3:	0.01058	0.11796	0.72786	0.84750	0.05196	0.24122	0.84257	0.91327	0.10067	0.32622	0.81563	0.89986
F4:	0.00632	0.09023	0.66336	0.80562	0.03649	0.19713	0.79688	0.88138	0.07181	0.27788	0.76779	0.87240
HC4:	0.00937	0.11191	0.71628	0.83875	0.04883	0.23481	0.83849	0.91119	0.09666	0.32177	0.81381	0.89924
Homo:	0.00880	0.10718	0.71272	0.84214	0.05361	0.24211	0.83891	0.91044	0.11395	0.34091	0.82043	0.90274

Table 5.8: Power of Test

$n = 100$	$k = 2$	Replication $= 1000 \times 100$											
α		0.01				0.05				0.10			
a	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00	0.10	1.00	5.00	10.00
F0:	0.00884	0.15058	0.75272	0.89116	0.05182	0.26367	0.83336	0.93959	0.11343	0.34589	0.84036	0.91340
HC0:	0.00889	0.15635	0.77784	0.90353	0.04875	0.26401	0.84470	0.94465	0.10069	0.33617	0.84455	0.91786
F1:	0.00813	0.14655	0.74995	0.88999	0.04919	0.25863	0.83168	0.93888	0.10890	0.34109	0.83884	0.91266
HC1:	0.00889	0.15635	0.77784	0.90353	0.04875	0.26401	0.84470	0.94465	0.10069	0.33617	0.84455	0.91786
F2:	0.00692	0.13779	0.74042	0.88487	0.04398	0.24605	0.82497	0.93562	0.09829	0.32643	0.83239	0.90869
HC2:	0.00859	0.15462	0.77570	0.90251	0.04783	0.26253	0.84391	0.94443	0.09935	0.33497	0.84413	0.91767
F3:	0.00543	0.12573	0.72766	0.87825	0.03694	0.23005	0.81573	0.93128	0.08482	0.30790	0.82430	0.90380
HC3:	0.00837	0.15296	0.77335	0.90174	0.04713	0.26103	0.84323	0.94420	0.09834	0.33378	0.84389	0.91752
F4:	0.00420	0.11199	0.70404	0.86515	0.03028	0.20893	0.79796	0.92258	0.06944	0.28120	0.80807	0.89292
HC4:	0.00784	0.14829	0.76723	0.89949	0.04563	0.25692	0.84187	0.94370	0.09622	0.33128	0.84322	0.91722
Homo:	0.00592	0.13073	0.73726	0.88378	0.04230	0.24375	0.82532	0.93592	0.09822	0.32748	0.83416	0.90982

Chapter 6

Conclusions

This dissertation considers the problems of binary classification with high-dimensional data, variable selection in sparse semiparametric single-index models and statistical inference under the presence heteroskedasticity.

We first compare the performance of popular existing machine learning methods for binary classification. We start by introducing four component-wise boosting methods, namely component-wise Discrete AdaBoost, component-wise Real AdaBoost, component-wise LogitBoost and component-wise Gentle AdaBoost. These methods are extremely popular since they are both computationally efficient and easy to implement. Moreover, the component-wise Boosting algorithms deal with high dimensional issue by considering the explanatory one at a time. In each iteration, only the most effective explanatory variable is chosen to train a weak learner. Hence, these methods allows $k \gg n$. However, hyper-parameters such as the number of boosting iteration normally need to be determined by the user prior to the estimation procedure. Cross-validation may also be used to choose the number of iterations. We conduct extensive comparison of the above mentioned methods through Monte Carlo experiments. We compare the methods using both traditional binary classification model (logistic

model) and irregular model (circle model). The boosting methods work well in both the traditional models and irregular models. We also use these methods for predicting the changing direction of the real personal income in the United States. The application show similar results as in the simulation of logistic models.

We next extend our study to the problem of binary classification under state-dependent loss functions. The state-dependent loss function in binary classification is often implicitly given in economics problems. We introduce a new Asymmetric AdaBoost algorithm which produces an additive regression model from maximizing a new risk function, namely the asymmetric exponential risk function. The new Asymmetric AdaBoost algorithm is based on the asymmetric exponential risk function, which maps into a binary decision making problem given a utility function. Furthermore, by carefully establishing the asymmetry in the risk function in accordance to the binary decision making, we show that our Asymmetric AdaBoost algorithm is closely related to the maximum score regression (Manski 1975, 1985) and the binary prediction literature in economics (Granger and Pesaran 2000, Lee and Yang 2006, Lahiri and Yang 2012, and Elliot and Lieli 2013), all of which however deal with low-dimensional predictor space. Asymmetric AdaBoost can handle the maximum score and binary prediction when the predictors are high-dimensional. Theoretical results show that Asymmetric AdaBoost will converge to Bayes risk as $n \to \infty$. Simulation results show that Asymmetric AdaBoost is a competitive approach in binary classification/prediction.

We then focus on the probability prediction of binary variables. We propose a new algorithm, based on the Rodeo, for variable selection and estimation for the sparse semiparametric linear single-index models by viewing the bandwidths as the inverse of the parameters which form the linear single index. The basic idea of the modified Rodeo algorithm for SIM (which we call SIM-Rodeo) is to view the local bandwidth selection as a variable selection in sparse semiparametric single

index model by amplifying the inverse of the bandwidths for relevant variables while keeping the inverse of the bandwidths of irrelevant variables relatively small. The SIM-Rodeo algorithm is greedy as it solves the locally optimal path choice at each stage which can also be shown to attain the asymptotic optimality in mean square error for sparse semiparametric single index models. The SIM-Rodeo method is able to distinguish truly relevant explanatory variables from noisy irrelevant variables and gives a "competitive" estimator for the model. In addition, the algorithm is fast to finish the greedy steps. We compare the SIM-Rodeo with a Lasso-type approach by Zeng et al (2012) for estimation and variable selection in SIM, which Zeng et al (2012) call SIM-Lasso. Our Monte Carlo simulation shows that SIM-Rodeo outperforms SIM-Lasso in variable selection and also in estimation. The new method is superior to the usual Lasso type penalty in estimation because SIM-Rodeo does not introduce bias from using the additive Lasso penalty and is computationally more efficient. Simulation results also show that the proposed SIM-Rodeo is consistent for variable selection and has smaller integrated mean squared errors than using SIM-Lasso.

Last but not least, we investigates the methods of statistical inference under the presence of heteroskedasticity of unknown form. The performance of different heteroskedasticity-consistent variance estimators are studied both under homoskedasticity and heteroskedasticity. We provide the analytical bias and variance of different heteroskedasticity-consistent variance estimators under homoskedasticity as well as numerical analysis. Under heteroskedasticity, we use the Imhof (1961) technique to approximate the distribution of the F-type test statistic using heteroskedasticity-consistent variance estimators and propose corrected methods for getting the correct critical values of the test statistic. Monte Carlo results show that the proposed method has better size and power than standard methods.

Bibliography

Allaire, J. J. and F. Chollet (2018). *keras: R Interface to 'Keras'*.

Angrist, J. and J.-S. Pishke (2009). *Mostly harmless econometrics*, Volume 53. Princeton University Press.

Bao, Y., A. Ullah, and Y. Wang (2017, oct). Distribution of the mean reversion estimator in the Ornstein–Uhlenbeck process. *Econometric Reviews 36*(6-9), 1039–1056.

Bliss, C. I. (1934a, jan). The method of probits. *Science 79*(2037), 38–39.

Bliss, C. I. (1934b). The method of probits. *Science 79*(2037), 409–410.

Bühlmann, P. (2006, apr). Boosting for high-dimensional linear models. *The Annals of Statistics 34*(2), 559–583.

Bühlmann, P. and B. Yu (2003, jun). Boosting with the L_2 loss: Regression and classification. *Journal of the American Statistical Association 98*(462), 324–339.

Chatterjee, S. (2016). *fastAdaboost: a Fast Implementation of Adaboost*.

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins (2018, jul). Double/debiased machine learning for treatment and structural parameters. *Econometrics Journal 21*(1), C1–C68.

Chesher, A. (1989, jul). Hajek Inequalities, Measures of Leverage and the Size of Heteroskedasticity Robust Wald Tests. *Econometrica 57*(4), 971.

Chu, J., T.-H. Lee, and A. Ullah (2018a). Asymmetric AdaBoost for High-Dimensional Maximum Score Regression.

Chu, J., T.-H. Lee, and A. Ullah (2018b). Variable Selection in Sparse Semiparametric Single Index Model.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological) 20*(2), 215–242.

Cribari-Neto, F. (2004, mar). Asymptotic inference under heteroskedasticity of unknown form. *Computational Statistics and Data Analysis 45*(2), 215–233.

Cribari-Neto, F. and M. D. G. A. Lima (2010, dec). Sequences of bias-adjusted covariance matrix estimators under heteroskedasticity of unknown form. *Annals of the Institute of Statistical Mathematics 62*(6), 1053–1082.

Culp, M., K. Johnson, and G. Michailidis (2016). *ada: The R Package Ada for Stochastic Boosting*.

Davidson, R. and E. Flachaire (2008). The wild bootstrap, tamed at last. *Journal of Econometrics 146*(1), 162–169.

Davidson, R. and J. G. MacKinnon (1993). *Estimation and inference in econometrics*. Oxford University Press.

Efron, B. and T. Hastie (2004). Least angle regression. *The Annals of statistics 32*(2), 407–499.

Eicker, F. (1963, jun). Asymptotic Normality and Consistency of the Least Squares Estimators for Families of Linear Regressions. *The Annals of Mathematical Statistics 34*(2), 447–456.

Elliott, G. and R. P. Lieli (2013, may). Predicting binary outcomes. *Journal of Econometrics 174*(1), 15–26.

Freund, Y. and R. Schapire (1996). Experiments with a New Boosting Algorithm. Technical report.

Freund, Y. and R. E. Schapire (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences 55*, 119–139.

Friedman, J., T. Hastie, and R. Tibshirani (2000a, apr). Additive logistic regression: A statistical view of boosting. *Annals of Statistics 28*(2), 337–407.

Friedman, J., T. Hastie, and R. Tibshirani (2000b, apr). Additive logistic regression: A statistical view of boosting. *Annals of Statistics 28*(2), 337–407.

Friedman, J., T. Hastie, and R. Tibshirani (2010a, feb). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software 33*(1), 1–22.

Friedman, J., T. Hastie, and R. Tibshirani (2010b). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software 33*(1).

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics 29*(5), 1189–1232.

Fritsch, S. and F. Guenther (2016). *neuralnet: Training of Neural Networks*.

Gaddum, J. (1933). Report on Biological Standards III: Methods of Biological Assay Depending on Quantal Response. *Special Report Series of the Medical Research Council 183*(London: Medical Research Council).

Granger, C. W. (1999). Outline of forecast theory using generalized cost functions. *Spanish Economic Review 1*(2), 161–173.

Granger, C. W. J. and M. H. Pesaran (2000, dec). Economic and statistical measures of forecast accuracy. *Journal of Forecasting 19*(7), 537–560.

Hamermesh, D. S. and J. E. Biddle (1994). Beauty and the Labor Market. *The American Economic Review 84*(5), 1174–1194.

Hansen, B. E. (2017). The Exact Distribution of the t-ratio with Robust and Clustered Standard Errors.

Hausman, J. and C. Palmer (2012, aug). Heteroskedasticity-robust inference in finite samples. *Economics Letters 116*(2), 232–235.

Hinkley, D. V. (1977, aug). Jackknifing in unbalanced situations. *Technometrics 19*(3), 285–292.

Huang, J., J. L. Horowitz, and F. Wei (2010, aug). Variable selection in nonparametric additive models. *Annals of Statistics 38*(4), 2282–2313.

Huber, P. (1967). The behavior of maximum likelihood estimates under nonstandard conditions. *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability 1*, 221–233.

Ichimura, H. (1993). Semiparametric {L}east {S}quares and {W}eighted {SLS} {E}stimation of {S}ingle {I}ndex {M}odels. *Journal of Econometrics 58*(1-2), 71–120.

Imbens, G. W. and M. Kolesár (2016, oct). Robust standard errors in small samples: Some practical advice. *Review of Economics and Statistics 98*(4), 701–712.

Imhof, J. P. (1961, dec). Computing the distribution of quadratic forms in normal variables. *Biometrika 48*(3 and 4), 419–426.

Jurado, K., S. C. Ludvigson, and S. Ng (2015). Measuring Uncertainty. *American Economic Review 105*(3), 1177–1216.

Klein, R. W. and R. H. Spady (1993, mar). An Efficient Semiparametric Estimator for Binary Response Models. *Econometrica 61*(2), 387.

Lafferty, J. and L. Wasserman (2008a, feb). Rodeo: Sparse, greedy nonparametric regression. *Annals of Statistics 36*(1), 28–63.

Lafferty, J. and L. Wasserman (2008b). Rodeo: Sparse, greedy nonparametric regression. *Annals of Statistics 36*(1), 28–63.

Lahiri, K. and L. Yang (2012, sep). Forecasting binary outcomes. In G. Elliott and A. Timmermann (Eds.), *Handbook of Economic Forecasting*, Volume 2, pp. 1025–1106. SSRN.

Lee, T. H. and Y. Yang (2006, nov). Bagging binary and quantile predictors for time series. *Journal of Econometrics 135*(1-2), 465–497.

MacKinnon, J. (2011). Thirty Years of Heteroskedasticity-Robust Inference.

MacKinnon, J. G. and H. White (1985a, sep). Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics 29*(3), 305–325.

MacKinnon, J. G. and H. White (1985b, sep). Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics 29*(3), 305–325.

Manski, C. F. (1975). Maximum score estimation of the stochastic utility model of choice. *Journal of Econometrics 3*(3), 205–228.

Manski, C. F. (1985). Semiparametric analysis of discrete response. Asymptotic properties of the maximum score estimator. *Journal of Econometrics 27*(3), 313–333.

McCaffrey, D. F. and R. M. Bell (2002). Bias Reduction in Standard Errors for Linear and Generalized Linear Models with Multi-Stage Samples. *Survey Methodology 28*(2), 169–181.

Mease, D., A. Wyner, and A. Buja (2007). Cost-weighted boosting with jittering and over/under-sampling: Jous-boost. *Journal of Machine Learning Research 8*, 409–439.

Nadaraya, E. A. (1964, jan). On Estimating Regression. *Theory of Probability & Its Applications 9*(1), 141–142.

Nakamura, A. and M. Nakamura (1998, mar). Model specification and endogeneity. *Journal of Econometrics 83*(1-2), 213–237.

Ng, S. (2014a, feb). Viewpoint: Boosting recessions. *Canadian Journal of Economics 47*(1), 1–34.

Ng, S. (2014b, feb). Viewpoint: Boosting recessions. *Canadian Journal of Economics 47*(1), 1–34.

Olson, M. (2017). *JOUSBoost: Implements Under/Oversampling for Probability Estimation*.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Rao, C. R. (1970). Estimation of Heteroscedastic Variances in Linear Models. *Journal of the American Statistical Association 65*(329), 161–172.

Rice, J. (1984, dec). Full-Text. *The Annals of Statistics 12*(4), 1215–1230.

Ridgeway, G. (2017). *gbm: Generalized Boosted Regression Models*.

Su, L. and Y. Zhang (2014, jan). Variable Selection in Nonparametric and Semiparametric Regression Models. In J. S. Racine, L. Su, and A. Ullah (Eds.), *The Oxford Handbook of Applied Nonparametric and Semiparametric Econometrics and Statistics*. Oxford University Press.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological) 58*(1), 267–288.

Tuszynski, J. (2018). *caTools: Tools: moving window statistics, GIF, Base64, ROC AUC, etc.*

Ullah, A. (2004). *Finite Sample Econometrics (Chap. 1,2)*. Oxford University Press.

Walker, S. H. and D. B. Duncan (1967, jun). Estimation of the probability of an event as a function of several independent variables. *Biometrika 54*(1), 167–179.

Watson, G. S. (1964). Smooth regression analysis. *The Indian Journal of Statistics 26*(4), 359–372.

White, H. (1980). A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity. *Econometrica 48*(4), 817.

Young, A. (2016). Improved, Nearly Exact, Statistical Inference with Robust and Clustered Covariance Matrices using Effective Degrees of Freedom Corrections.

Zellner, A. (1986, jun). Bayesian estimation and prediction using asymmetric loss functions. *Journal of the American Statistical Association 81*(394), 446–451.

Zeng, P., T. He, and Y. Zhu (2012). A lasso-type approach for estimation and variable selection in single index models. *Journal of Computational and Graphical Statistics 21*(1), 92–109.

Zou, H. (2006). The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association 101*(476), 1418–1429.

Appendix A

Proofs for Chapter 3

A.1 Proof of Theorem 1

Proof. Let $F^* = \arg\min_F R_\tau(F)$ be the Bayes classifier. Let $\mathcal{P}(x,y)$ be the joint density function of x and y, and $P_w(x,y) = \frac{t(y,x)\mathcal{P}(x,y)}{\int t(y,x)\mathcal{P}(x,y)dydx}$. Then $P_w(x,y)$ defines a probability distribution of (x,y) on $\chi \times \{\pm 1\}$. By definition,

$$\begin{aligned}
R_{\psi,\tau}(F_i) &= \mathbb{E}\left(t(y,x)\, e^{-yF_i}\right) \\
&= \int t(y,x)\, e^{-yF_i} \mathcal{P}(x,y)\, dydx \\
&= \int t(y,x)\, \mathcal{P}(x,y)\, dydx \cdot \int e^{-yF_i} \frac{t(y,x)\, \mathcal{P}(x,y)}{\int t(y,x)\, \mathcal{P}(x,y)\, dydx} dydx \\
&= \int t(y,x)\, \mathcal{P}(x,y)\, dydx \cdot \int e^{-yF_i} P_w(x,y)\, dydx \\
&= C \int e^{-yF_i} P_w(x,y)\, dydx,
\end{aligned}$$

where $C \equiv \int t(y,x)\, \mathcal{P}(x,y)\, dydx$ is positive and bounded. Moreover,

$$R^*_{\psi,\tau} = \inf_{F_i} R_{\psi,\tau}(F_i).$$

Hence, by Theorem 2,

$$R_{\psi,\tau} \to R^*_{\psi,\tau} \quad \text{implies that} \quad \int 1_{(y \neq \text{sign}[F_i])} P_w(x,y)\, dydx \to \int 1_{(y \neq \text{sign}[F_i])} P_w(x,y)\, dydx.$$

Rewrite the expression in terms of $\mathcal{P}(x,y)$, we have

$$\frac{1}{C} \int 1_{(y \neq \text{sign}[F_i])} t(y,x)\, \mathcal{P}(x,y)\, dydx \to \frac{1}{C} \int 1_{(y \neq \text{sign}[F^*])} t(y,x)\, \mathcal{P}(x,y)\, dydx.$$

Therefore,

$$R_\tau(\text{sign}[F_i]) = \int t(y,x)\, 1_{(y \neq \text{sign}[F_i])} \mathcal{P}(x,y)\, dydx \to \int t(y,x)\, 1_{(y \neq \text{sign}[F^*])} \mathcal{P}(x,y)\, dydx = R^*_\tau.$$

□

A.2 Proof of Theorem 2

We start with the asymmetric exponential risk function

$$R_{\psi,\tau}(F(x)) = \mathbb{E}\left(t(y,x)\, e^{-yF(x)}\right). \tag{A.1}$$

Step 1 is to look for the optimal $f_{m+1}(x)$ for each iteration. Suppose we have finished m iterations, the current classifier is denoted as $F_m(x) = \sum_{s=1}^m c_s f_s(x)$. In the next iteration, we are seeking an update $c_{m+1} f_{m+1}(x)$ for the function fitted by previous iterations $F_m(x)$. The updated classifier would be

$$F_{m+1}(x) = F_m(x) + c_{m+1} f_{m+1}(x). \tag{A.2}$$

The risk for the updated classifier is

$$R_{\psi,\tau}\left(F_m\left(x\right)+c_{m+1}f_{m+1}\left(x\right)\right)=\mathbb{E}\left(t\left(y,x\right)e^{-y(F_m(x)+c_{m+1}f_{m+1}(x))}\right). \tag{A.3}$$

Expand it w.r.t. $f_{m+1}\left(x\right)$

$$\begin{aligned}
& \mathbb{E}\left(t\left(y,x\right)e^{-y(F_m(x)+c_{m+1}f_{m+1}(x))}\right) && \text{(A.4)}\\
\approx\ & \mathbb{E}\left(t\left(y,x\right)e^{-yF_m(x)}\left(1-yc_{m+1}f_{m+1}\left(x\right)+\frac{yc_{m+1}^2f_{m+1}^2\left(x\right)}{2}\right)\right) && \text{(A.5)}\\
=\ & \mathbb{E}\left(t\left(y,x\right)e^{-yF_m(x)}\left(1-yc_{m+1}f_{m+1}\left(x\right)+\frac{c_{m+1}^2}{2}\right)\right), && \text{(A.6)}
\end{aligned}$$

since $y^2=f_{m+1}^2\left(x\right)=1$ holds for all y and $f_{m+1}\left(x\right)$. Only the second term in the bracket contains $f_{m+1}\left(x\right)$, so minimizing the above risk function w.r.t. $f_{m+1}\left(x\right)$ is equivalent to maximizing the following expectation

$$\max_f \mathbb{E}\left(e^{-yF_m(x)}t\left(y,x\right)yf_{m+1}\left(x\right)|x\right), \tag{A.7}$$

for any $c_{m+1}>0$. Let weights be $w\equiv e^{-yF_m(x)}$. Then we re-write the above maximization as

$$\max_f \mathbb{E}_w\left(t\left(y,x\right)yf_{m+1}\left(x\right)|x\right). \tag{A.8}$$

Solve the maximization problem

$$\begin{aligned}
& \max_f \mathbb{E}_w\left(t\left(y,x\right)yf_{m+1}\left(x\right)|x\right) && \text{(A.9)}\\
=\ & P_w\left(y=1|x\right)t\left(1,x\right)f_{m+1}\left(x\right)-P_w\left(y=-1|x\right)t\left(-1,x\right)f_{m+1}\left(x\right) && \text{(A.10)}\\
=\ & \left[P_w\left(y=1|x\right)t\left(1,x\right)-P_w\left(y=-1|x\right)t\left(-1,x\right)\right]\ f_{m+1}\left(x\right), && \text{(A.11)}
\end{aligned}$$

$f_{m+1}(x)$ should take the same sign as $P_w\left(y=1|x\right)t\left(1,x\right)-P_w\left(y=-1|x\right)t\left(-1,x\right)$.

The solution is

$$f_{m+1}(x) = \begin{cases} 1, & P_w(y=1|x)\, t(1,x) - P_w(y=-1|x)\, t(-1,x) > 0 \\ -1, & \text{otherwise.} \end{cases} \tag{A.12}$$

Step 2 is to look for the optimal c_{m+1} for each iteration. After solving $f_{m+1}(x)$, we minimize the risk function (A.3) w.r.t. c_{m+1},

$$\begin{aligned} c_{m+1} &= \arg\min_c R_{\psi,\tau}\left(F_m(x) + c f_{m+1m+1}(x)\right) && \text{(A.13)} \\ &= \arg\min_c \mathbb{E}\left(t(y,x)\, e^{-y(F_m(x)+c_{m+1}f_{m+1}(x))}\right) && \text{(A.14)} \\ &= \arg\min_c \mathbb{E}_w\left(t(y,x)\, e^{-y c_{m+1} f_{m+1}(x)}\right) && \text{(A.15)} \end{aligned}$$

Then

$$\begin{aligned} & \mathbb{E}_w\left(t(y,x)\, e^{-y c_{m+1} f_{m+1}(x)}\right) && \text{(A.16)} \\ = \;& P_w(y=1, f_{m+1}(x)=1)\, t(1,x)\, e^{-c_{m+1}} + P_w(y=-1, f_{m+1}(x)=-1)\, t(-1,x)\, e^{-c_{m+1}} && \text{(A.17)} \\ & + P_w(y=1, f_{m+1}(x)=-1)\, t(1,x)\, e^{c_{m+1}} + P_w(y=-1, f_{m+1}(x)=1)\, t(-1,x)\, e^{c_{m+1}} && \text{(A.18)} \end{aligned}$$

The first order condition from taking the derivative w.r.t. c_{m+1}

$$\begin{aligned} & \frac{\partial R_{\psi,\tau}(c_{m+1} f_{m+1}(x))}{\partial c_{m+1}} && \text{(A.19)} \\ = \;& -P_w(y=1, f_{m+1}(x)=1)\, t(1,x)\, e^{-c_{m+1}} - P_w(y=-1, f_{m+1}(x)=-1)\, t(-1,x)\, e^{-c_{m+1}} && \text{(A.20)} \\ & + P_w(y=1, f_{m+1}(x)=-1)\, t(1,x)\, e^{c_{m+1}} + P_w(y=-1, f_{m+1}(x)=1)\, t(-1,x)\, e^{c_{m+1}} && \text{(A.21)} \end{aligned}$$

gives the optimal c_{m+1} from solving the following

$$P_w\left(y=1, f_{m+1}\left(x\right)=1\right) t\left(1, x\right) e^{-c_{m+1}} + P_w\left(y=-1, f_{m+1}\left(x\right)=-1\right) t\left(-1, x\right) e^{c_{m+1}} \tag{A.22}$$

$$= P_w\left(y=1, f_{m+1}\left(x\right)=-1\right) t\left(1, x\right) e^{c_{m+1}} + P_w\left(y=-1, f_{m+1}\left(x\right)=1\right) t\left(-1, x\right) e^{-c_{m+1}} \tag{A.23}$$

where $P_w\left(y=1, f_{m+1}\left(x\right)=1\right)$ is the rate of true positive (TP), $P_w\left(y=-1, f_{m+1}\left(x\right)=-1\right)$ is the rate of true negative (TN), $P_w\left(y=1, f_{m+1}\left(x\right)=-1\right)$ is the rate of false negative (FN), $P_w\left(y=-1, f_{m+1}\left(x\right)=1\right)$ is the rate of false positive (FP). Hence, rewriting it as

$$\left[\mathrm{TP} \times t\left(1, x\right) + \mathrm{TN} \times t\left(-1, x\right)\right] e^{-c_{m+1}} = \left[\mathrm{FN} \times t\left(1, x\right) + \mathrm{FP} \times t\left(-1, x\right)\right] e^{c_{m+1}}, \tag{A.24}$$

we obtain the optimal c_{m+1}

$$c_{m+1} = \frac{1}{2} \log\left(\frac{\mathrm{TP} \times t\left(1, x\right) + \mathrm{TN} \times t\left(-1, x\right)}{\mathrm{FN} \times t\left(1, x\right) + \mathrm{FP} \times t\left(-1, x\right)}\right) = \frac{1}{2} \log\left(\frac{1 - err_{m+1}}{err_{m+1}}\right), \tag{A.25}$$

where $err_{m+1} = \mathbb{E}_w\left(t\left(y, x\right) \times 1_{(y \neq f_{m+1}(x))}\right)$.

Step 3 is to update the current strong learner and get ready for the next iteration. In the next iteration, we have

$$F_{m+1}\left(x\right) \leftarrow F_m\left(x\right) + c_{m+1} f_{m+1}\left(x\right). \tag{A.26}$$

Hence

$$w_{m+1} = e^{-yF_{m+1}(x)} \tag{A.27}$$

$$= e^{-y(F_m(x) + c_{m+1} f_{m+1}(x))} \tag{A.28}$$

$$= w_m \times e^{-c_{m+1} y f_{m+1}(x)}, \tag{A.29}$$

is of identical form as in Algorithm 2. □

A.3 Proof of Theorem 3

Hereby we provide the proof of Theorem 3 following Bartlett and Traskin (2007) for our asymmetric exponential risk.

Notation. Let $R_{\psi,\tau,n}$ be the sample version of $R_{\psi,\tau}$ with sample size n and the set of k-combinations, $k \in \mathbb{N}$, of functions in $\mathcal{H}$

$$\mathcal{F}^k = \left\{ F | F = \sum_{i=1}^{k} \lambda_i h_i, \lambda_i \in \mathbb{R}, h_i \in \mathcal{H} \right\}. \tag{A.30}$$

Define the squashing function $\pi_l(\cdot)$ to be

$$\pi_l(x) = \begin{cases} l, & x > l \\ x, & x \in [-l, l] \\ -l, & x < -l. \end{cases} \tag{A.31}$$

Then the set of truncated functions is

$$\pi_l \circ \mathcal{F} = \left\{ \tilde{F} | \tilde{F} = \pi_l(F), F \in \mathcal{F} \right\}. \tag{A.32}$$

The set of classifiers based on a class $\mathcal{F}$ is denoted by

$$\mathcal{G} = \{ G(F) | F \in \mathcal{F} \}. \tag{A.33}$$

Let

$$\varphi_\lambda = \inf_{\alpha \in [-\lambda, \lambda]} t(y, x) e^{-\alpha}. \tag{A.34}$$

Assumptions 1. *Let n be sample size. Let there exist non-negative sequences $M_n \to \infty$, $\zeta_n \to \infty$ and a sequence $\left\{\bar{F}_n\right\}_{n=1}^{\infty}$ of reference functions such that*

$$R_{\psi,\tau}\left(\bar{F}_n\right) \underset{n\to\infty}{\to} R^*_{\psi,\tau}, \tag{A.35}$$

and suppose that the following conditions are satisfied.

1. *Uniform convergence of M_n-combinations.*

$$\sup_{F\in\pi_{\zeta_n}\circ\mathcal{F}^{M_n}} \left|R_{\psi,\tau}(F) - R_{\psi,\tau,n}(F)\right| \overset{a.s.}{\underset{n\to\infty}{\to}} 0. \tag{A.36}$$

2. *Convergence of empirical exponential risks for the sequence $\left\{\bar{F}_n\right\}_{n=1}^{\infty}$.*

$$\max\left\{0, R_{\psi,\tau,n}\left(\bar{F}_n\right) - R_{\psi,\tau}\left(\bar{F}_n\right)\right\} \overset{a.s.}{\underset{n\to\infty}{\to}} 0. \tag{A.37}$$

3. *Algorithmic convergence of M_n-combinations.*

$$\max\left\{0, R_{\psi,\tau,n}\left(F_{M_n}\right) - R_{\psi,\tau,n}\left(\bar{F}_n\right)\right\} \overset{a.s.}{\underset{n\to\infty}{\to}} 0. \tag{A.38}$$

Evidence that Asymmetric AdaBoost and the asymmetric exponential risk satisfies Assumption 1 can be find in Bartlett and Traskin (2007) where they discuss the same topic for symmetric AdaBoost.

Proof of Theorem 3. We follow the procedure of Bartlett and Traskin (2007) with our asymmetric exponential risk function. For almost every outcome ω on the probability space we can define sequences $\epsilon_n^1(\omega) \to 0$, $\epsilon_n^2(\omega) \to 0$,$\epsilon_n^3(\omega) \to 0$, such that for almost

all ω the following inequalities are true.

$$\begin{aligned} R_{\psi,\tau}\left(\pi_{\zeta_n}\left(F_{M_n}\right)\right) &\leq R_{\psi,\tau,n}\left(\pi_{\zeta_n}\left(F_{M_n}\right)\right)+\epsilon_n^1(\omega)\to 0 \text{ by } (A.36) \\ &\leq R_{\psi,\tau,n}\left(F_{M_n}\right)+\epsilon_n^1(\omega)+\varphi_{\zeta_n} \qquad (A.39) \\ &\leq R_{\psi,\tau,n}\left(\bar{F}_{M_n}\right)+\epsilon_n^1(\omega)+\varphi_{\zeta_n}+\epsilon_n^2(\omega) \text{ by } (A.38) \\ &\leq R_{\psi,\tau}\left(\bar{F}_n\right)+\epsilon_n^1(\omega)+\varphi_{\zeta_n}+\epsilon_n^2(\omega)+\epsilon_n^3(\omega) \text{ by } (A.37) \qquad (A.40) \end{aligned}$$

Inequality (A.39) follows from the convexity of $\varphi(\cdot)$. By choice of the sequence $\left\{\bar{F}_n\right\}_{n=1}^{\infty}$, we have $R_{\psi,\tau}\left(\bar{F}_n\right)\to R_{\psi,\tau}^*$ and $\varphi_{\zeta_n}\to 0$. And from (A.40) follows

$$R_{\psi,\tau}\left(F_{M_n}\right)\overset{a.s.}{\to} R_{\psi,\tau}^*. \qquad (A.41)$$

Hence, Asymmetric AdaBoost is consistent if stopped after M_n steps. □

Appendix B

Proofs for Chapter 4

We follow the notation of Lafferty and Wasserman (2008b) and write $Y_n = \tilde{O}_P(a_n)$ to mean that $Y_n = O_P(b_n a_n)$ where b_n is logarithmic in n. And we write $a_n = \Omega(b_n)$ if $\liminf_n \left|\frac{a_n}{b_n}\right| > 0$ and $a_n = \tilde{\Omega}(b_n)$ if $a_n = \Omega(b_n c_n)$ where c_n is logarithmic in n.

Define

$$\mu_j(\theta) = \frac{\partial}{\partial \theta_j} E\left[\hat{m}_\theta(x) - m(x) \,|X_1, \ldots, X_n\right],$$

which is the derivative of the conditional bias. The first lemma analyzes $\mu_j(\theta)$ and $\mathbb{E}(\mu_j(\theta))$ under the assumption that f is uniform. The second lemma analyzes the variance. The third lemma bounds the probabilities $\mathbb{P}(|Z_j| \geq \lambda_j)$ in terms of tail inequalities for standard normal variables.

In each of these lemmas, we make the following assumptions. We assume that f is uniform, K is a Gaussian kernel, and $\alpha > 1$. Moreover, without loss of generality, we make use of the following set $\mathcal{B}$ of coefficients where $\theta_0 > 0$

$$\mathcal{B} = \left\{ \theta = (\theta_1, \ldots, \theta_k) = \left(\underbrace{\alpha^{t_1}\theta_0, \ldots, \alpha^{t_r}\theta_0}_{r \text{ terms}}, \underbrace{\theta_0, \ldots, \theta_0}_{k-r \text{ terms}} \right) : 0 \leq t_j \leq T_n, j = 1, \ldots, r \right\},$$

where $T_n \leq c_1 \log n$. Finally, we assume that

$$\begin{aligned} r &= O(1), \\ k &= O\left(\frac{\log n}{\log \log n}\right), \\ \theta_0 &= c_0 \log \log n. \end{aligned}$$

The proofs of the lemmas can be found in Lafferty and Wasserman (2008b).

Lemma 1. *For each $\theta \in \mathcal{B}$,*

$$\mathbb{E}(\mu_j(\theta)) = \begin{cases} \frac{\nu_2 m_{jj}(x\theta)}{\theta_j} + \frac{g_j(x_R\theta_R)}{\theta_j}, & j \leq r, \\ 0, & j > r, \end{cases}$$

where $\nu_2 I = \int uu^T K(u)\, du$ and $g_j(x_R\theta_R)$ depends only on the relevant variables and bandwidths, and satisfies

$$|g_j(x_R\theta_R)| = O\left(\sum_{l \leq r} \sup_x \frac{|m_{jjll}(x\theta)|}{\theta_l^2}\right).$$

Furthermore, for any $\delta > 0$,

$$\Pr\left(\max_{\substack{\theta \in \mathcal{B} \\ 1 \leq j \leq k}} \frac{|\mu_j(\theta) - \mathbb{E}(\mu_j(\theta))|}{s_{j(\theta)}} > \frac{\sqrt{\delta \log n}}{\log \log n} \leq \frac{1}{n^{\delta\sigma^2/(8c_0)}}\right)$$

where

$$s_j^2(\theta) = \frac{C\theta_j^2}{n} \prod_{l=1}^{k} \theta_l,$$

with

$$C = \sigma^2 \frac{\int K^2(u)\, du}{f(x)}.$$

Lemma 2. *Let* $\nu_j(\theta) = \mathrm{Var}(Z_j|X_1,\ldots,X_n)$. *Then*

$$\Pr\left(\max_{\substack{\theta\in\mathcal{B}\\1\le j\le k}}\left|\frac{\nu_j(\theta)}{s_j^2(\theta)}-1\right|>\varepsilon\right)\to 0,$$

for all $\varepsilon > 0$.

Lemma 3. *For any* $c > 0$ *and each* $j > r$,

$$\Pr\left(|Z_j(\theta_0)| > \lambda_j(\theta_0)\right) = o\left(\frac{1}{n^c}\right).$$

Uniformly for $\theta \in \mathcal{B}$, $c > 0$ *and* $j \le r$,

$$\Pr\left(|Z_j(\theta)| < \lambda_j(\theta)\right) \le \Pr\left(N(0,1) > \frac{\nu_j\left|m_{jj}(x\theta)\right| + z_n}{s_j(\theta)\,\theta_j}\right) + o\left(\frac{1}{n^c}\right),$$

where $z_n = O\left(\theta_j^{-3}\right)$.

Proof of Theorem 1. Let $\mathcal{A}_t$ be the active set at step t. Define S_t to be the event that $\mathcal{A}_t = \{1,\ldots,r\}$. We want to show that

$$\Pr(S_1) \to 1,$$

from which the theorem follows.

Fix $c > 0$. In what follows, we let $\xi_n(c)$ denote a term that is $o(n^{-c})$; we will suppress the dependence on c and simply write ξ_n.

At Step $t = 1$, define the event

$$B_1 = \{|Z_j| > \lambda_j \text{ for all } j \le r\} \cap \{|Z_j| < \lambda_j \text{ for all } j > r\}.$$

Thus, $A_1 = B_1$. We claim that

$$\Pr\left(B_1^c\right) \leq O\left(\frac{1}{n}\right) + \xi_n.$$

From Lemma 3, when $j > r$,

$$\Pr\left(\max_{j>r} |Z_j| > \lambda_j\right) \leq \sum_{j=r+1}^{k} \Pr\left(|Z_j| > \lambda_j\right) \leq d\xi_n = \xi_n.$$

When $j \leq r$,

$$\Pr\left(|Z_j| < \lambda_j \text{ for some } j \leq r\right) \leq O\left(\frac{1}{n}\right) + \xi_n.$$

Hence,

$$\Pr\left(\theta_j = \theta_0 \text{ for all } j > r\right) \to 1 \text{ as } n \to \infty.$$

and

$$\Pr\left(\theta_j > \theta_0 \text{ for all } j \leq r\right) \to 1 \text{ as } n \to \infty$$

□