GRADIENT BOOSTING MACHINE FOR HIGH-DIMENSIONAL ADDITIVE MODELS

A DISSERTATION
SUBMITTED TO THE INSTITUTE FOR
COMPUTATIONAL AND MATHEMATICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Tong Xia
June 2014

This dissertation is online at: http://purl.stanford.edu/qw734xz4236

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Tse Lai, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael Saunders, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mei-Chiung Shih

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

The gradient boosting machine proposed by Friedman (2001) represents a fundamental advance in statistical learning. Although previous studies have shown the effectiveness of this algorithm on fitting additive models when the sample size is much bigger than the number of covariates, its properties in the high-dimensional case, where the number of covariates exceeds the sample size, are still an open area of research. We extend the application of the gradient boosting machine to a high-dimensional censored regression problem, and use simulation studies to show that this algorithm outperforms the currently used iterative regularization method.

Bühlmann (2006) proved the convergence and consistency of the gradient boosting machine for high-dimensional linear models. We establish the same asymptotic theory for nonlinear additive models by showing that the set of basis functions is highly redundant. Then we discuss the properties of the 10-fold cross validation, which can be used in practice to define a criterion for choosing the number of iterations for the gradient boosting machine. We illustrate the effectiveness of the gradient boosting machine plus this stopping criterion on fitting high-dimensional additive models through numerical examples.

Acknowledgements

I feel extremely grateful for my advisor, Professor Tze Leung Lai. His knowledge and patience greatly inspired me during the entire course of my Ph.D. study. He helped me choose this nice topic, and guided me to exploit the whole field of statistical learning. I have benefited immensely from his thorough and insightful advice. By working with him, I have not only acquired expertise, but also learned how to analyze and solve complex problems, which will be tremendously helpful for my future work. Besides academic study, he also gave me lots of valuable suggestion on career development and life. I feel so lucky and honored to be his student.

I would like to thank members of my oral exam committee, Professor Michael Saunders, Professor Mei-Chiung Shih, Professor George Papanicolaou and Professor Marc Coram. In particular, I will thank Professor Saunders and Professor Shih, who also served on my reading committee, for spending time reading my thesis and for providing valuable comments and feedbacks.

I would also like to thank the directors of ICME, Professor Walter Murray, Professor Peter Glynn and Professor Margot Gerritsen, for leading a nice program that provides us with an enormous amount of opportunities of doing academic research and working in the high-tech industry. I also want to thank our student service manager, Indira Choudhury, for her kind help on my work and life during the whole period of my graduate study.

I'm grateful for my colleagues and friends at Stanford. Among them are Tania Bakhos, Chao Chen, Xing Fu, Xiaoye Jiang, Dongwoo Kim, Xiangrui Meng, Dai Shi, Kai Wai Tsang, Zhiyu Wang, Tailai Wen, Bangpeng Yao, Ernest Yu, Hongsong Yuan, Xianyi Zeng, Xin Zhou and Chenhao Zhu. This is, however, a list that is far

from complete. I benefited a lot by discussing with them on various problems about academic research and career development. Besides, they brought me lots of joy and happiness in my spare time. I want to thank all of them for making my life meaningful and colorful at Stanford.

It is to my dear wife, WANG Muge and my parents, XIA He and WANG Min that I owe the deepest gratitude. They gave me the courage to overcome all the obstacles along this journey. It was their confidence in me and in my work that enabled me to bring my doctorate to fruition.

In the end, I dedicate this dissertation to my grandfather, XIA Kui, who passed away last year. June 13, 2014 is his 90th birthday. My dissertation and doctorate degree will be my best gifts for him, which I believe he would be pleased to see in the Heaven. I hope that the love and respect I feel for him shines through on every page.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the development of technologies, it is now more and more convenient to acquire large amounts of data. The time of Big Data is coming! How to take advantage of the data and extract useful information out of it is now a hot area of research. In particular, people want to predict (or "learn") something unknown from existing data. For example, based on a user's web browsing history, a company that provides online services wants to predict if he or she will click the link for some news article, or purchase some merchandise. Based on the location, size, number of bedrooms and the year of building of a house, a real estate agent wants to predict the price of the house if it is put on the market. It is very easy to list dozens of examples like this. The technique of prediction can be applied to various areas.

The mathematical abstract of the prediction problem is like this: Suppose we have a system consisting of a random output variable Y and random input vector $\mathbf{X} = (X_1, \ldots, X_p)$, which satisfy the following relationship:

$$Y = f(\mathbf{X}) + \varepsilon, \tag{1.1}$$

where ε is a random variable that is independent of $\mathbf{X}$, has mean 0 and small variance, and $f(\cdot)$ is some unknown function, or *statistical model*. Our goal is to learn the model $f(\cdot)$ so that given some input vector $\mathbf{x}$, we can predict the corresponding outcome y as $f(\mathbf{x})$. Through our discussion, we use bold face fonts to represent vectors. Also, we

use upper case letters to represent random vectors (variables) and lower case letters to represent determined values.

We also define a nonnegative loss function $L(y, f(\mathbf{x}))$ that measures the discrepancy between the real output y and the predicted value $f(\mathbf{x})$. The most frequently used loss function is the squared loss

$$L(y, f(\mathbf{x})) = \frac{1}{2}(y - f(\mathbf{x}))^2. \tag{1.2}$$

There are other formats of loss functions as well, such as the absolute loss

$$L(y, f(\mathbf{x})) = |y - f(\mathbf{x})| \tag{1.3}$$

and the exponential loss

$$L(y, f(\mathbf{x})) = \log(1 + \exp(f(\mathbf{x}))) - yf(\mathbf{x}). \tag{1.4}$$

The ideal model f^* is the one that minimizes the expected loss. That is,

$$f^* = \arg\min_f \mathbf{E}L(Y, f(\mathbf{X})), \tag{1.5}$$

where the expectation is taken in the joint distribution $\mathcal{D}$ of $\mathbf{X}$ and Y.

However, in practice, $\mathcal{D}$ is usually difficult or impossible to know. So there is no way to solve the minimization problem (1.5) explicitly. Instead, people observe a finite set of samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, which are generated from $\mathcal{D}$. These samples are called the *training data*, or *training samples*. The unknown distribution $\mathcal{D}$ can then be approximated by the empirical distribution of these training samples. Therefore, instead of solving problem (1.5), we solve the following approximated problem:

$$\hat{f} = \arg\min_f \frac{1}{n} \sum_{t=1}^{n} L(y_t, f(\mathbf{x}_t)). \tag{1.6}$$

The solution $\hat{f}(\cdot)$ will be used as the the model for future prediction.

However, even with known, finite set of training samples, it is still difficult to

solve the minimization problem (1.6), since the set of models $f(\cdot)$ is too large. Based on the prior knowledge of the data, people can assume that $f(\cdot)$ has some special structure. In general, $f(\cdot)$ is restricted to be a number of a parametric class of functions $\{f(\mathbf{x};\mathbf{p})\}$. In this case, problem (1.6) reduces to the problem of solving for the optimal parameters $\mathbf{p}$ from the training samples $(\mathbf{x}_t, y_t)$, $t = 1, \ldots, n$.

1.1 Additive Models and the Gradient Boosting Machine

In this thesis, we focus on the case where the model $f(\cdot)$ can be represented by a linear expansion of a series of basis functions. That is, we assume that the output Y and input $\mathbf{X}$ have the following relationship:

$$Y = \sum_{k=1}^{N} \beta_k \phi(X_{j_k}; r_k) + \varepsilon, \tag{1.7}$$

where

$$\{X_{j_k} : k = 1, \ldots, N\} = \{X_1, \ldots, X_p\}.$$

The function $\phi(X_{j_k}; r_k)$ is a *basis function*. In general, it is a nonlinear function of the j_k-th covariate X_{j_k} with nonlinear parameter r_k. For this additive model, the set of parameters $\mathbf{p}$ includes the number of basis functions N, all the linear coefficients β_k and nonlinear parameters r_k, which need to be estimated from the training samples. So our goal is to solve the following minimization problem:

$$\min_{N, \{\beta_j, r_j\}_1^N} \sum_{t=1}^{n} L(y_t, \sum_{k=1}^{N} \beta_k \phi(x_{tj_k}; r_k)). \tag{1.8}$$

One commonly used basis function for the nonlinear model is the indicator function

$$\phi(X_{j_k}; r_k) = \mathbf{1}_{\{X_{j_k} \leqslant r_k\}} \text{ or } \phi(X_{j_k}; r_k) = \mathbf{1}_{\{X_{j_k} \geqslant r_k\}}.$$

This corresponds to the *tree* model. The tree model can be applied to various fields such as clinical trials, where sometimes people only care about whether some indices of a patient are above (or below) some thresholds and are not interested in their exact values.

The bases function for the tree model is non-differentiable, and sometimes is approximated by the smooth sigmoid function

$$\phi(X_{j_k}; \mathbf{r}_k) = \frac{\exp(a_k + b_k X_{j_k})}{1 + \exp(a_k + b_k X_{j_k})}.$$

This corresponds to a simplified version of the *neural network* model. Here the nonlinear parameter $\mathbf{r}_k$ is a 2D vector (a_k, b_k).

One of the techniques that have been widely used in fitting the above additive models is Boosting, which is one of the most successful techniques in the machine learning community. The idea of boosting is to combine a series of "weak learners" to generate a powerful estimator. Due to the flexibility of choosing the expression and number of "weak learners", boosting is well suited to various modeling problems.

According to Hastie et al. (2009), "boosting is a way of fitting an additive expansion in a set of elementary basis functions". Assume that the model has the form (1.7). In general, the basis functions $\phi(X_{j_k}; r_k)$ have very simple forms, like the indicator or sigmoid function as we have shown before. Each basis function by itself could be a very poor model for the original problem, so they are called "weak learners". However, by adding these basis functions iteratively, boosting is able to generate a powerful model with very good prediction performance.

The concept of "basis function" has different names under different contexts. In machine learning, it is called the *base learner*, while in signal processing, the set of all basis functions is called the *dictionary*. In all the cases, the basis functions are the basic units of the model and represent the set of all possible search directions during the boosting iteration.

One of the most popular boosting algorithms is *AdaBoost*, which was first introduced by Freund and Schapire (1997). The AdaBoost algorithm is designed for classification problems, where the output Y is limited in a finite set. The procedure

of this algorithm is to iteratively apply a series of weak classifiers $G_1(\cdot), \ldots, G_m(\cdot)$ to modified versions of the training data, then output a linear combination of these weak classifiers as the final model. Here are the details of the AdaBoost algorithm.

Algorithm 1 AdaBoost

Initialize the weights $w_i = \frac{1}{n},\ i = 1, \ldots, n$.

for k *= 1, . . . , m* ***do***

Fit a (weak) classifier $G_k(\mathbf{x})$ *to the training samples* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ *using weights* w_i.

Compute $err_k = \frac{\sum_{i=1}^{n} w_i \mathbf{1}_{\{y_i \neq G_k(\mathbf{x}_i)\}}}{\sum_{i=1}^{n} w_i}$.

Compute $\alpha_k = \log((1 - err_k/err_k))$.

Update the weights $w_i \leftarrow w_i \cdot \exp(\alpha_k \cdot \mathbf{1}_{\{y_i \neq G_k(\mathbf{x}_i)\}}),\ i = 1, \ldots, n$.

end for

Output $G(\mathbf{x}) = sign[\sum_{k=1}^{m} \alpha_k G_k(\mathbf{x})]$.

During the k-th iteration, the algorithm updates the weight for each training sample so that samples that are wrongly classified by the classifier $G_k(\cdot)$ receive higher weights. When the iteration completes, the algorithm outputs a linear combination of all the weak classifiers $G_1(\cdot), \ldots, G_m(\cdot)$, where the coefficient of each classifier is based on its prediction performance on the training samples. Various examples have shown that the final classifier generated by AdaBoost could be very powerful despite the weakness of the original classifiers $G_1(\cdot), \ldots, G_m(\cdot)$.

It can be shown that when $Y \in \{-1, 1\}$, the AdaBoost iterations are equivalent to solving the following minimization problems:

$$(\alpha_k, G_k) = \arg\min_{\alpha, G} \frac{1}{n} \sum_{t=1}^{n} \exp[-y_t(\sum_{j=1}^{k-1} \alpha_j G_j(\mathbf{x}_t) + \alpha G(\mathbf{x}_t))],\ k = 1, \ldots, m.$$

In other words, during each iteration, the AdaBoost algorithm adds a new classifier $G_k(\mathbf{x})$ to the existing model

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{k-1} \alpha_j G_j(\mathbf{x})$$

so that the loss function

$$L(y, f(\mathbf{x})) = \exp(-yf(\mathbf{x})) \tag{1.9}$$

is minimized. Therefore, the AdaBoost algorithm is trying to solve the minimization problem (1.8) with loss function (1.9).

As discussed before, there are other forms of loss functions. For example, the squared loss (1.2) is widely used in various regression problems. In (binary) classification problems, besides (1.9), the exponential loss (1.4) is also a commonly used loss function.

In order to solve the minimization problem (1.8) for general loss and basis functions (other than the binary classifier G), Friedman (2001) proposed the *gradient boosting machine*, which adds basis functions to the model iteratively based on the "steepest descent" principle. Given the current model $\hat{f}^{k-1}(\mathbf{x})$, the gradient boosting machine adds in the k-th iteration a new basis function $\phi(x_{\hat{j}_k}; b_k)$ that is as close as possible to the negative gradient (the steepest descent direction) $-\frac{\partial L}{\partial f}\big|_{f=\hat{f}^{k-1}}$ and then chooses the corresponding coefficient $\hat{\rho}_k$, i.e. the step size in the direction $\phi(x_{\hat{j}_k}; b_k)$, by minimizing the loss function. Here is a sketch of the gradient boosting machine.

Algorithm 2 Gradient Boosting Machine

$\hat{f}^0(\mathbf{x}) = 0$.

for $k = 1, \ldots, m$ ***do***

$\hat{u}_t^{k-1} = -\frac{\partial L}{\partial f}(y_t, \hat{f}^{k-1}(\mathbf{x}_t))$.

$(\hat{j}_k, \hat{b}_k) = \arg\min_{j,b} \frac{1}{n}\sum_{t=1}^n [\hat{u}_t^{k-1} - \beta\phi(x_{tj}; b)]^2 = \arg\max_{1\leqslant j\leqslant p, b\in\mathbb{R}} \frac{|\frac{1}{n}\sum_{t=1}^n \hat{u}_t^{k-1}\phi(x_{tj};b)|}{\sqrt{\frac{1}{n}\sum_{t=1}^n \phi(x_{tj};b)^2}}$.

$\hat{\rho}_k = \arg\min_\rho \frac{1}{n}\sum_{t=1}^n L(y_t, \hat{f}^{k-1}(\mathbf{x}_t) + \rho\phi(x_{t\hat{j}_k}; \hat{b}_k))$.

$\hat{f}^k(\mathbf{x}) = \hat{f}^{k-1}(\mathbf{x}) + \hat{\rho}_k\phi(\mathbf{x}; \hat{b}_k)$.

end for

Output $\hat{f}^m(\cdot)$.

It can be seen that the gradient boosting machine does not have a limit on the format of loss or basis functions. Therefore, it can be applied to fit various models. We next illustrate its application to some common regression and classification problems. For each of the examples we give the specific form of the algorithm.

Linear Model

The linear model is one of the most important statistical models. It assumes that Y is a linear function of $\mathbf{X}$, i.e.

$$Y = \alpha + \mathbf{X}\boldsymbol{\beta} + \varepsilon = \alpha + \sum_{j=1}^{p} \beta_j X_j + \varepsilon. \tag{1.10}$$

This model has a very simple form, and can be applied to various practical problems. Also, in many cases nonlinear models can be transformed such that the input and output have a linear relationship. In addition, the technique for solving the linear model is the foundation for studying more complex models.

For linear models, we use squared loss (1.2) as the loss function. Supposing we have training samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, we want to solve for

$$(\hat{\alpha}, \hat{\boldsymbol{\beta}}) = \arg\min_{\alpha,\boldsymbol{\beta}} \frac{1}{n} \sum_{t=1}^{n} (y_t - \sum_{j=1}^{p} \beta_j x_j - \alpha)^2.$$

Let $\tilde{\boldsymbol{\beta}} = (\alpha, \boldsymbol{\beta})^T$ be the augmented vector with first entry α, and $\tilde{\mathbf{x}} = (1, \mathbf{x})^T$ be the augmented vector with first entry 1. Then the minimization problem is equivalent to

$$\hat{\boldsymbol{\beta}} = \arg\min_{\tilde{\boldsymbol{\beta}}} \frac{1}{n} \sum_{t=1}^{n} (y_t - \sum_{j=1}^{p+1} \tilde{\beta}_j \tilde{x}_j)^2.$$

Therefore, we will assume $\alpha = 0$ in the sequel. So our purpose is to solve

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{t=1}^{n} (y_t - \sum_{j=1}^{p} \beta_j x_j)^2. \tag{1.11}$$

The linear model (1.10) can be treated as a special case of the additive model (1.7) with basis function

$$\phi(x_j; r_j) = x_j.$$

Here the nonlinear parameter r_j can be viewed as zero-dimensional. When the loss function is defined as the squared loss (1.2), the gradient boosting machine is also

called the L_2 *boosting algorithm*. In the L_2 boosting algorithm, the negative gradient $\hat{u}_t^{k-1}$ reduces to the residual $y_t - \hat{f}_{k-1}(\mathbf{x}_t)$. Also, the step size $\hat{\rho}_k$ is the same as $\hat{\beta}_k$, so there is no need to do a line search after the new basis function $x_{\hat{j}_k}$ is identified. In each step, the algorithm picks the covariate $x_{\hat{j}_k}$ that has the largest absolute sample correlation with the current residual $\hat{r}^{k-1}(\mathbf{x}) = y - \hat{f}^{k-1}(\mathbf{x})$, then does a univariate least squares regression of $\hat{r}^{k-1}$ on $x_{\hat{j}_k}$, and adds $x_{\hat{j}_k}$ along with the regression coefficient to the current model $\hat{f}^{k-1}(\mathbf{x})$, then updates the residual. It can be seen that the L_2 boosting algorithm chooses the basis functions and regression coefficients in a greedy way. It regresses the current residual on the most correlated covariates without taking other covariates into account. Therefore, it is also called the *pure greedy algorithm* (PGA).

Regression Tree

The regression tree assumes that Y is the summation of indicator functions of $\mathbf{X}$:

$$Y = \sum_{k=1}^{N} \beta_i \mathbf{1}_{\{X_{j_k} \in R_k\}} + \varepsilon.$$

Here the basis function is $\phi(x_{j_k}; r_k) = \mathbf{1}_{\{x_{j_k} \in R_k\}}$. The nonlinear parameter r_k corresponds to the interval R_k that contains X_{j_k}. For continuous valued X_{j_k}, R_k could be an unbounded real interval $(-\infty, r_k]$, $[r_k, \infty)$, or some bounded interval $[r_k^{(1)}, r_k^{(2)}]$. For discrete (or finite) valued X_{j_k}, R_k could be the set of one or several possible values of X_{j_k}.

If the loss function is squared loss, then the negative partial derivative at the k-th iteration is the residual

$$\hat{r}_t^{k-1} = y_t - \sum_{i=1}^{k-1} \hat{\beta}_k \mathbf{1}_{\{x_{t\hat{j}_i} \in R_i\}}.$$

For the next basis function $\mathbf{1}_{\{x_{t\hat{j}_k} \in R_k\}}$, the algorithm searches over all possible indices of $\mathbf{x}$ and corresponding intervals such that $\mathbf{1}_{\{x_{t\hat{j}_k} \in R_k\}}$ is most correlated with $\hat{r}_t$. That

is,

$$(\hat{j}_k, R_k) = \arg\max_{1\leqslant j\leqslant p,R} \frac{|\frac{1}{n}\sum_{t=1}^{n} \hat{r}_t^{k-1}\mathbf{1}_{\{x_{tj}\in R\}}|}{\sqrt{\frac{1}{n}\sum_{t=1}^{n}\mathbf{1}^2_{\{x_{tj}\in R\}}}} = \arg\max_{1\leqslant j\leqslant p,R} \frac{|\frac{1}{n}\sum_{x_{tj}\in R}\hat{r}_t^{k-1}|}{\sqrt{\frac{1}{n}\#\{t|x_{tj}\in R\}}}.$$

The coefficient $\hat{\beta}_k$ is the regression coefficient of $\hat{r}_t$ on $\mathbf{1}_{\{x_{t\hat{j}_k}\in R_k\}}$, which reduces to the mean of $\hat{r}_t$'s where $x_{t\hat{j}_k} \in R_k$:

$$\hat{\beta}_k = \text{mean}\{\hat{r}_t | x_{t\hat{j}_k} \in R_k\}.$$

Logistic Regression

In the logistic regression model, the output variable Y is binary.

$$Y \sim \text{Bernoulli}(\frac{\exp(f(\mathbf{X}))}{1+\exp(f(\mathbf{X}))}),$$

where $f(\cdot)$ is an additive function of $\mathbf{X}$:

$$f(\mathbf{X}) = \sum_{k=1}^{N} \beta_k \phi(X_{j_k}; r_k).$$

Here we use the exponential loss (1.4).

Starting with $\hat{f}^0(\cdot) = 0$, let $\hat{f}^{k-1}(\cdot)$ be the model obtained at the $(k-1)$-th iteration. The gradient boosting machine calculates $\hat{u}_t^{k-1}$ as

$$\hat{u}_t^{k-1} = y_t - \frac{\exp(\hat{f}^{k-1}(\mathbf{x}_t))}{1+\exp(\hat{f}^{k-1}(\mathbf{x}_t))}.$$

The next basis function $\phi(x_{\hat{j}_k}; \hat{r}_k)$ is the one that is most correlated with $\hat{u}^{k-1}$, and the coefficient $\hat{\rho}_k$ is defined as

$$\hat{\rho}_k = \arg\min_{\rho} \frac{1}{n}\sum_{t=1}^{n}[\log(1+\exp(\hat{f}^{k-1}(\mathbf{x}_t) + \rho x_{t\hat{j}_k})) - y_t(\exp(\hat{f}^{k-1}(\mathbf{x}_t) + \rho x_{t\hat{j}_k}))].$$

Then the model $\hat{f}^{k-1}(\mathbf{x})$ is then updated to

$$\hat{f}^k(\mathbf{x}) = \hat{f}^{k-1}(\mathbf{x}) + \hat{\rho}_k \phi(x_{\hat{j}_k}; \hat{r}_k).$$

Suppose the algorithm stops after m iterations. Given an input vector $\mathbf{x}$, outcome y is predicted as

$$\hat{y} = \mathbf{1}_{\{\hat{f}^m(\mathbf{x}) \geqslant 0\}}.$$

Friedman (1999) used numerical examples to illustrate the performance of the gradient boosting machine when the sample size n is greater than the number of covariates p. It turns out that this algorithm is very efficient. It converges to the real model very fast with small computational cost. In addition, its form is very flexible, and could be applied to various linear and nonlinear models.

Friedman (1999) considers the case when the nonlinear parameter $\mathbf{r}$ in the basis function $\phi(x_{j_k}; \mathbf{r}_k)$ is finite-dimensional. In fact, the gradient boosting machine could also be extended to infinite-dimensional nonlinear parameters. This could be applied to problems where some of the outcome y's in the training data are censored. Instead of observing the true survival time y_i for each sample i, we actually observe the pair (t_i, δ_i), where $t_i = \min\{y_i, c_i\}$. Here c_i is some value that "cut" y off and $\delta_i = \mathbf{1}_{\{y_i \leqslant c_i\}}$ is the indicator of whether y_i is censored by c_i. In general, it is assumed that c_i is independent of the input vector $\mathbf{x}_i$. In this censored case, the problem of fitting the model is coupled with imputing the censored outcome y. The distribution of y is then represented by an infinite-dimensional nonlinear parameter. In Chapter 2, we implement the gradient boosting machine to combine the procedures of selecting basis functions and estimating the distribution of the outcome y. This process, followed by an *orthogonal greedy algorithm*, has desired performance in both variable selection and data imputation.

1.2 Convergence and Consistency of Gradient Boosting Machine in the High-Dimensional Case

In the low-dimensional case where the number of covariates p is smaller than the sample size n, many problems can be solved efficiently without using the gradient boosting machine. For example, in the linear regression problem, the minimizer of (1.11) has a closed form. Let X be the $n \times p$ data matrix whose i-th row equals $(x_{i1}, \ldots, x_{ip})$, $i = 1, \ldots, n$, and $\mathbf{y} = (y_1, \ldots, y_n)^T$ be the $n \times 1$ vector of outputs. Then the minimization problem (1.11) can be transformed to the vector form

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2. \tag{1.12}$$

Here we have ignored the coefficient $\frac{1}{n}$. The optimizer $\hat{\boldsymbol{\beta}}$ can be solved from the normal equation:

$$X^T X\hat{\boldsymbol{\beta}} = X^T\mathbf{y}. \tag{1.13}$$

When $p < n$, the matrix $X^T X$ has full rank, so the solution $\hat{\boldsymbol{\beta}}$ is defined and unique. This is *ordinary least squares regression*. It can be shown that $\hat{\boldsymbol{\beta}}$ is an unbiased estimator of the coefficient for the real underlying model. That is,

$$\mathbf{E}\hat{\boldsymbol{\beta}} = \boldsymbol{\beta},$$

where the expectation is taken on the training samples, and $\boldsymbol{\beta}$ is the minimizer of $\mathbf{E}(Y - \mathbf{X}\boldsymbol{\beta})^2$.

However, in recent years, more and more people become interested in the high-dimensional case, where the number of covariates p far exceeds the sample size n. This high-dimensional problem is very common in areas such as biology and clinical trials, where it is possible to get the information for thousands of genes for each sample, thanks to the development bioinformatics technology, while there are only hundreds, or even dozens of samples available, since they are very expensive.

Ordinary least squares regression fails to solve the problem when $p \gg n$. First,

in this high-dimensional case, the matrix $X^T X$ is not of full rank, and the normal equation (1.13) will have infinitely many solutions. Also, although the dimension of the model is very high, in most cases only a few covariates are relevant to the output y, i.e. with corresponding coefficients nonzero. In other words, the model is highly *sparse*. Therefore, besides prediction, one also needs to select the correct set of relevant variables (the variables with non-zero regression coefficients). But the normal equation (1.13) does not give any hint on the relevance of variables.

In contrast, PGA does not have a limit on n and p, so it can be applied to high-dimensional problems. Also, since PGA selects variables according to their correlation with the current residual, it has a high tendency for choosing relevant variables. Bülhmann (2006) studied the asymptotic properties of PGA in the high-dimensional case. He proved that under some regularity and sparsity conditions,

$$\mathbf{E}|\hat{f}^m(\mathbf{X}) - f(\mathbf{X})|^2 \to 0 \tag{1.14}$$

as $n \to +\infty$ and $m = m_n \to +\infty$ sufficiently slowly. Here $f(\mathbf{x}) = \sum_{j=1}^{p} \beta_j x_j$ is the true regression function. $\hat{f}^m(\mathbf{x})$ is the model fitted by PGA after m iterations, and the expectation is taken on the random input vector $\mathbf{X}$.

Another class of methods for fitting high-dimensional linear models is *regularization methods*. The idea is to add a penalty function to the minimization problem (1.11) to make the solution unique and shrink the regression coefficients toward zero. We go over more details of regularization methods and compare their performance to the gradient boosting machine in the following chapters.

For nonlinear models (1.7), the problem of high-dimensionality is becoming more and more common as well. The convergence and consistency of the gradient boosting machine in this case is still an open area of research. Recall that for the linear model, the set of basis functions is finite (i.e., all the covariates $x_1, \ldots, x_p$). However, for nonlinear models, the nonlinear parameters r_k come from a continuous set. This makes the set of basis functions (the dictionary) infinite (actually uncountable), thereby adding more difficulties in choosing the correct basis function during each iteration and bounding the distance to the real model. In Chapter 3, we address

this problem by dividing all the basis functions into finite groups and bounding the difference within each group, which allows us to view the set of basis functions as finite. We then extend Bülhmann's result by establishing the asymptotic theory of the gradient boosting machine on nonlinear models. This is the main contribution of the thesis.

The asymptotic theory verifies the convergence and consistency of the gradient boosting machine when both the sample size n and number of iterations m go to infinity. However, in practice, with a fixed number of samples, we would not iterate too much, because that will make the model to be too complex and may also affect its prediction performance. Instead, we need to stop the iteration as soon as we've made enough progress through the boosting procedure. One of the most commonly used methods for choosing the optimal number of iterations (or other tuning parameters) is cross validation. In Chapter 4, we discuss the principle of cross validation and illustrate its performance through simulation study.

1.3 Outline of the Thesis

The thesis is organized as follows. In Chapter 2, we extend the gradient boosting machine to problems with censored outcomes. Here we propose a two-stage algorithm. In the first stage, we apply the gradient boosting machine (PGA) to select the variable and estimate the distribution of the current residual alternately so that these two processes can benefit from each other. At the end of this stage, we obtain consistent imputation of the output y's. In the next step, we apply an orthogonal greedy algorithm (OGA) to the imputed data to refine the model so that irrelevant variables are eliminated. We then use numerical examples to illustrate the performance of our algorithm. We will see that it is more accurate and efficient than regularization methods.

In Chapter 3, we establish the asymptotic theory of the gradient boosting machine for general additive models. First, we study the convergence and consistency of the algorithm in an "ideal" case, where the joint distribution of the input $\mathbf{X}$ and output Y

is known. Then we show that in the practical case, when we have only a finite number of training samples, the outcome of the gradient boosting machine is "close" enough to the outcome of the "ideal case" algorithm, and we conclude that the "practical" outcome will also converge to the real model.

In Chapter 4, we illustrate the performance of the gradient booting machine through simulation studies. In the first section we discussion cross validation as a general rule of estimating the prediction error and choosing the tuning parameter for various statistical modeling algorithms. Then in the following sections we implement the gradient boosting machine with cross validation on numerical examples. We will see the efficiency of this algorithm on general additive sparse models.

We summarize our contributions and conclusions in Chapter 5.

Chapter 2

Gradient Boosting Machine on Censored Data

In this chapter we study the problem where some of the outputs y's in the training samples are censored. Censoring is a very common phenomenon in areas such as clinical trials, where people want to study the relationship between the survival time of patients and their gene expression data, and some patients' survival times are not fully observed, since they may fail to follow up or survive beyond the period of the experiment. Therefore, instead of observing the true survival time y_i for each patient i, we actually observe $t_i = \min(y_i, c_i)$, where c_i is the time of censoring (e.g. the time that the patient fails to follow up or the ending time of the experiment). Besides, we also observe the indicator $\delta_i = \mathbf{1}_{\{y_i \leqslant c_i\}}$ for each sample i. In other words, we know whether each of the survival times is censored.

For each sample i, we fully observe the input vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$, which corresponds to the gene expression data of the i-th patient. We assume that the input $\mathbf{X}$ is independent of the time of censoring C and we want to learn the relationship between $\mathbf{X}$ and the true survival time Y. In this chapter, we first go over the traditional technique for the low-dimensional case where p is less than n. Then we consider the reverse case, and review regularization methods that addresses high-dimensionality. As discussed in the previous chapter, the censored regression problem can be viewed

as an additive modeling problem with infinite-dimensional nonlinear parameters. We propose a two-stage algorithm that is composed of the gradient boosting machine (PGA) and the orthogonal greedy algorithm (OGA). In the end of this chapter, we use simulation studies to illustrate the performance of our two-stage algorithm and compare it to regularization methods.

2.1 Traditional Technique for the $p < n$ Case

In the classical case where the sample size n is greater than number of covariates p, there are two types of models solving the censored regression problem: The *Proportional Hazards Model* and *the Accelerated Failure Time Model*. We briefly go over the basic idea for each of them.

2.1.1 The Proportional Hazards Model

The Proportional Hazards Model was proposed by Cox (1972). It assumes that the probability of dying (the "hazard") is proportional to some exponential function of the input vector. Let Y be the (random) survival time. Define survival function of Y as

$$S(t) = \mathbf{P}(Y \geqslant t) = 1 - F(t-), \tag{2.1}$$

where

$$F(t) = \mathbf{P}(Y \leqslant t)$$

is the cumulative distribution function of Y and

$$F(t-) = \lim_{s \to t-} F(s) = \mathbf{P}(Y < t).$$

Assume $S(t)$ is differentiable. Then the probability density function of Y is

$$f(t) = -\frac{dS(t)}{dt}.$$

Define the *hazard function* as

$$\lambda(t) = \lim_{h \to 0+} \frac{\mathbf{P}\big(t \leqslant y \leqslant t+h\big)}{h} = \frac{f(t)}{S(t)}. \tag{2.2}$$

Integrating with respect to t, we have

$$S(t) = \exp[-\int_0^t \lambda(u)du] = \exp(-\Lambda(t)), \tag{2.3}$$

where $\Lambda(t) = \int_0^t \lambda(s)ds$ is called the *cumulative hazard function*. Given observed data $\{(t_i, \delta_i)|i = 1, \ldots, n\}$, let

$$Y(s) = \sum_{i=1}^{n} \mathbf{1}_{\{t_i \geqslant s\}} \tag{2.4}$$

be the number of samples that are at risk at time s, and

$$N(s) = \sum_{i=1}^{n} \mathbf{1}_{\{t_i \leqslant s, \delta_i = 1\}} \tag{2.5}$$

be the number of samples that didn't survive beyond time s. Then the cumulative hazard function $\Lambda(t)$ can be estimated by

$$\hat{\Lambda}(t) = \sum_{s \leqslant t} \frac{\Delta N(s)}{Y(s)} = \int_0^t \frac{\mathbf{1}_{\{Y(s)>0\}}}{Y(s)} dN(s), \tag{2.6}$$

where $\Delta N(s) = N(s) - N(s-)$ is the number of observed failures at time s. $\hat{\Lambda}(t)$ is a piecewise constant function where jumps happen only at non-censored failure times. If there is no censoring, then $\hat{\Lambda}(t)$ would be the empirical distribution of the sample y_i's.

Cox (1972) proposed the Proportional Hazards Model that relates the hazard function $\lambda(t)$ to the input vector $\mathbf{x}$ through the formula

$$\lambda(t) = \lambda_0(t) \exp(\boldsymbol{\beta}^T \mathbf{x}), \tag{2.7}$$

where $\lambda_0(t)$ is called the *baseline hazard*. In order to build the model (2.7), both the vector $\boldsymbol{\beta}$ and the baseline hazard $\lambda_0(\cdot)$ need to be estimated.

The vector $\boldsymbol{\beta}$ can be estimated through maximum partial likelihood. Let $y_{(1)} < \cdots < y_{(m)}$ be the ordered uncensored failure times with $m \leqslant n$. Define C_j as the set of censored survival t_i's in the interval $[y_{(j-1)}, y_{(j)})$ and (j) as the individual failing at $y_{(j)}$. Let $R_{(j)} = \{i : t_i \geqslant y_{(j)}\}$ be the set of samples that are at risk at time $y_{(j)}$. Then we have

$$\mathbf{P}((j) \text{ fails at } y_{(j)} | R_{(j)}, \text{ one failure at } y_{(j)}) = \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_{(j)})}{\sum_{i \in R_{(j)}} \exp(\boldsymbol{\beta}^T \mathbf{x}_{(i)})}.$$

Define the likelihood

$$\prod_{j=1}^{m} \mathbf{P}((j) \text{ fails at } y_{(j)} | R_{(j)}, \text{ one failure at } y_{(j)}).$$

Here the production is taken only on uncensored samples, so it is called the *partial likelihood*. Define $l(\boldsymbol{\beta})$ as the *partial log-likelihood*:

$$l(\boldsymbol{\beta}) = \sum_{j=1}^{m} (\boldsymbol{\beta}^T \mathbf{x}_{(j)} - \log(\sum_{i \in R_{(j)}} \exp(\boldsymbol{\beta}^T \mathbf{x}_{(i)}))). \tag{2.8}$$

The Cox regression estimator $\hat{\boldsymbol{\beta}}$ is defined as the maximizer of $l(\boldsymbol{\beta})$. It can be shown that $\hat{\boldsymbol{\beta}}$ is a consistent estimator of $\boldsymbol{\beta}$. In particular, $(l''(\hat{\boldsymbol{\beta}}))^{-1/2}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})$ converges to a standard normal vector as $n \to +\infty$, where $l''(\hat{\boldsymbol{\beta}})$ is the Hessian matrix of $l(\boldsymbol{\beta})$.

Let $\Lambda_0(t) = \int_0^t \lambda_0(s) ds$ be the *cumulative baseline hazard*. $\Lambda_0(\cdot)$ could be estimated through (2.6), with $Y(s)$ being redefined as

$$Y(s) = \sum_{i \in R_{(j)}} \exp(\boldsymbol{\beta}^T \mathbf{x}_i) \text{ for } s \in [y_{(j)}, y_{(j+1)}).$$

The estimator $\hat{\lambda}_0(\cdot)$ of $\lambda_0(\cdot)$ can be estimated by taking the derivative of $\Lambda_0(\cdot)$. If $\Lambda_0(\cdot)$ is piecewise constant, $\hat{\lambda}_0(\cdot)$ will be a linear combination of delta functions.

2.1.2 The Accelerated Failure Time Model

The second type of model is the *Accelerated Failure Time Model*. Here the (transformed) failure time is modeled as a linear function of the covariates.

$$g(Y) = \alpha + \beta^T \mathbf{X} + \varepsilon, \tag{2.9}$$

where $g(\cdot)$ is a monotone function, (e.g. $g(y) = \exp(y)$), ε is the random noise, has mean zero and finite variance, and is independent of the input vector $\mathbf{X}$.

If the output Y can be fully observed, we can apply ordinary least squares regression to estimate α and β. On the other hand, when Y is censored, Buckley and James (1979) propose to impute it by its expectation conditioned on $\mathbf{X}$ and the censoring time. For simplicity, assume the model (2.9) is

$$Y = \alpha + \boldsymbol{\beta}^T \mathbf{X} + \varepsilon.$$

Define the imputed failure time as

$$y_i^* = \begin{cases} y_i & \delta_i = 1, \\ \mathbf{E}(T \big| T > c_i, \mathbf{x}_i) & \delta_i = 0. \end{cases} \tag{2.10}$$

Let $\xi_i = t_i - \beta^T \mathbf{x}_i = \min\{y_i, c_i\} - \beta^T \mathbf{x}_i$ Then ξ_i is independent of x_i, and therefore

$$\begin{aligned} \mathbf{E}(T_i \big| T_i > y_i, \mathbf{x}_i) &= \boldsymbol{\beta}^T \mathbf{x}_i + \mathbf{E}(\xi_i \big| \xi_i > y_i - \boldsymbol{\beta}^T \mathbf{x}_i) \\ &= \boldsymbol{\beta}^T \mathbf{x}_i + \int_{Y_i - \boldsymbol{\beta}^T \mathbf{x}_i}^{\infty} \frac{t dF(t)}{1 - F(y_i - \boldsymbol{\beta}^T \mathbf{x}_i)}, \end{aligned} \tag{2.11}$$

where F is the distribution function of $\xi = T - \boldsymbol{\beta}^T \mathbf{X}$. Buckley and James (1979) estimated F through a non-parametric technique introduced by Kaplan and Meier (1958). Basically, let $y_{(1)} < \cdots < y_{(m)}$ be the ordered uncensored failure times. Let $y_{(0)} = 0$. For $j = 1, \ldots, m$, let n_j be the number of samples at risk at $y_{(j-1)}$, d_j be the number of failures during the interval $(y_{(j-1)}, y_{(j)}]$, and l_j be the number of samples

that were censured during the interval $(y_{(j-1)}, y_{(j)}]$. Then F is estimated by

$$1 - \hat{F}(t) = \prod_{j:y_{(j)} \leqslant t} (1 - \frac{d_j}{n_j - l_j}). \tag{2.12}$$

This is equivalent to the estimator (2.6) for the cumulative hazard. Again, $\hat{F}(t)$ is piecewise constant.

Once we have the estimator $\hat{F}(\cdot)$, we can estimate y_i^* by using (2.10) and (2.11). The coefficient vector $\boldsymbol{\beta}$ is then solved from the normal equation

$$X^{*T} X^* \hat{\boldsymbol{\beta}} = X^{*T} \mathbf{y}^*, \tag{2.13}$$

where $\mathbf{y}^* = (y_1^*, \ldots, y_n^*)^T$ and $X^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*)^T$ is the centered design matrix with

$$\mathbf{x}_i^* = \mathbf{x}_i - \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j = \mathbf{x}_i - \bar{\mathbf{x}}.$$

Note that $\mathbf{y}^*$ is actually a function of $\boldsymbol{\beta}$, (2.13) has to be solved iteratively. That is, we need to start with an initial guess of $\boldsymbol{\beta}$. During each iteration, we use the current estimated $\boldsymbol{\beta}$ to impute $\mathbf{y}^*$, and solve for a new estimator of $\boldsymbol{\beta}$ from (2.13). As the iteration goes on, the estimated $\boldsymbol{\beta}$ could either converge to some vector $\boldsymbol{\beta}^*$, which will be the solution of (2.13), or oscillate between two vectors. Buckley and James (1979) showed that when oscillation occurs, the two bounding values are generally close to each other, so any value between them (e.g. their arithmetic average) can be used to approximate the solution of (2.13). When (2.13) is solved, the intercept α is estimated by

$$\hat{\alpha} = \bar{\mathbf{y}}^* - \boldsymbol{\beta}^T \bar{\mathbf{x}}.$$

Buckley and James (1979) established the consistency of $\hat{\beta}$, but they claimed that the intercept α cannot be consistently estimated in general due to censoring. However, Wang et al. (2008)'s simulation results show that α can be estimated consistently when some of covariates are supported in the entire space $\mathbb{R}$.

2.2 Regularization Methods for the High-Dimensional Case

Just as the non-censored problem, people are getting more and more interested in the high-dimensional case where p is much larger than n. In this case the Proportional Hazards Model fails to have a consistent solution, so we focus on the Accelerated Failure Time model instead. Here the normal equation (2.13) has infinitely many solutions, since the matrix $X^{*T}X^*$ is not of full rank. Currently, this high-dimensionality is addressed by using regularization methods. In this section we first give a brief review for regularization methods on non-censored linear models and then discuss how they can be applied to the censored case.

2.2.1 Regularization Methods for High-Dimensional Linear Models

Recall that in order to fit a linear model (1.10), we need to solve the minimization problem (1.11). If all (or most) of the coefficients of the solution are nonzero, then there will be no way to solve this problem when $p > n$, since there is too little information and too many unknowns. So in general it is assumed that the model (1.10) is highly sparse; i.e. only a few coefficients are non-zero. Usually, this assumption makes sense in practice. For example, in clinical trials, although people have the data for thousands of genes, in most cases only dozens of them are really related to the disease they want to study. However, people don't know which variables are relevant (i.e. have nonzero regression coefficients), so they need some technique to select a few relevant variables out of thousands of candidates and fit an accurate model on them.

Regularization methods are a class of methods that successfully solve high-dimensional linear regression problems. The idea of regularization methods is to add a penalty function $P(\boldsymbol{\beta})$ to the optimization problem (1.11). Instead of trying to solve (1.11) when $p \gg n$, the methods are solving

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + P(\boldsymbol{\beta}). \tag{2.14}$$

The function $P(\cdot)$ is non-negative and increases with the magnitude of $\boldsymbol{\beta}$. The purpose of adding this penalty function is to make the solution of the optimization problem unique and to shrink the coefficients toward zero to enforce variable selection. Next we go over the most commonly used penalty functions and discuss their properties.

Ridge Regression

The first regularization method to be reviewed is *ridge regression*, which was introduced by Hoerl & Kennard (1970). Ridge regression uses the L_2 penalty $P(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_2^2$, where $\|\cdot\|_2$ represents the L_2 norm, and $\lambda > 0$ is the *tuning parameter* that determines the amount of shrinkage.

The estimator can be solved explicitly as

$$\hat{\boldsymbol{\beta}}^{Ridge} = (X^T X + \lambda \mathbf{I})^{-1} X^T \mathbf{y}.$$

It is easy to show that when $\lambda > 0$, the matrix $X^T X + \lambda \mathbf{I}$ is non-singular. Therefore, the above formulation is valid.

Ridge regression has two effects on the inputs as well as the estimator of the regression coefficients. The first effect is shrinkage. When the matrix X has orthogonal columns, it can be seen that

$$\hat{\boldsymbol{\beta}}^{Ridge} = \frac{1}{1+\lambda}\hat{\boldsymbol{\beta}}^{OLS}.$$

In the general case, consider the singular value decomposition (SVD) of X:

$$X = UDV^T,$$

where U and V are $n \times p$ and $p \times p$ orthogonal matrices that span the column space and row space of X respectively. D is a $p \times p$ diagonal matrix with diagonal elements $d_1 \geqslant d_2 \geqslant \cdots \geqslant d_p \geqslant 0$ representing the singular values of X. It is not hard to see

that

$$X\hat{\boldsymbol{\beta}}^{Ridge} = UD(D^2 + \lambda\mathbf{I})^{-1}VU^T\mathbf{y} = \sum_{j=1}^{p} \mathbf{u}_j \frac{d_j^2}{d_j^2 + \lambda}\mathbf{u}_j^T\mathbf{y}$$

and

$$X\hat{\boldsymbol{\beta}}^{OLS} = UU^T\mathbf{y} = \sum_{j=1}^{p} \mathbf{u}_j\mathbf{u}_j^T\mathbf{y}.$$

Compared to ordinary least squares regression, ridge regression shrinks the coordinates of the output with respect to D toward zero. The smaller the singular value d_j, the larger the amount of shrinkage that would be applied to the corresponding column. In the high-dimensional case, such shrinkage is usually necessary in recognition of the sparsity of the regression coefficients. Of course, when the shrinkage is applied, $\hat{\boldsymbol{\beta}}^{Ridge}$ will no longer be an unbiased estimator of $\boldsymbol{\beta}$. But the sacrifice in bias is usually worthwhile even when $n > p$ because it could result in significant reduction in variance. A proper choice of the tuning parameter λ would achieve good bias-variance tradeoff and the resulting model will have better prediction performance than the OLS model.

The second effect of ridge regression is *decorrelation*. To see this effect, assume that the design matrix X has been normalized so that each column has sample mean 0 and sample variance 1. Then

$$\frac{1}{n}X^TX = \begin{pmatrix} 1 & \rho_{12} & \cdots & \rho_{1p} \\ \rho_{21} & 1 & & \\ \vdots & & \ddots & \rho_{p-1,p} \\ \rho_{p1} & \cdots & \rho_{p,p-1} & 1 \end{pmatrix},$$

where $\rho_{ij} = \frac{1}{n}\sum_{t=1}^{n} x_{ti}x_{tj}$ is the sample covariance of variables x_i and x_j. Then

$$(X^TX + \lambda\mathbf{I})^{-1} = \frac{1}{n+\lambda}\begin{pmatrix} 1 & \frac{n}{n+\lambda}\rho_{12} & \cdots & \frac{n}{n+\lambda}\rho_{1p} \\ \frac{n}{n+\lambda}\rho_{21} & 1 & & \\ \vdots & & \ddots & \frac{n}{n+\lambda}\rho_{p-1,p} \\ \frac{n}{n+\lambda}\rho_{p1} & \cdots & \frac{n}{n+\lambda}\rho_{p,p-1} & 1 \end{pmatrix}^{-1}.$$

It's seen that the correlations are shrunk by a factor of $\frac{n}{n+\lambda}$, so the covariates appear to be "less correlated" than in the OLS case.

Lasso

Although ridge regression shrinks the coefficients toward 0, it doesn't make any of them to be exactly zero. So the estimator $\hat{\beta}^{Ridge}$ is usually not sparse and the resulting model usually does not have good interpretation if p is large. To obtain a sparse estimator, Tibshirani (1996) introduces the *Least Absolute Shrinkage and Selection Operator* (Lasso), which uses the L_1 norm $P(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$ as the penalty term. The corresponding optimization problem is

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1. \tag{2.15}$$

In general, the above problem has no closed form solution, but when the design matrix X has orthonormal columns, i.e. $X^TX = I$, it can shown that

$$\hat{\boldsymbol{\beta}}^{Lasso} = \begin{cases} 0, & |\hat{\boldsymbol{\beta}}_j^{OLS}| \leqslant \frac{\lambda}{2}, \\ \hat{\boldsymbol{\beta}}_j^{OLS} - \frac{\lambda}{2}, & \hat{\boldsymbol{\beta}}_j^{OLS} > \frac{\lambda}{2}, \\ \hat{\boldsymbol{\beta}}_j^{OLS} + \frac{\lambda}{2}, & \hat{\boldsymbol{\beta}}_j^{OLS} < -\frac{\lambda}{2}. \end{cases}$$

Hence the lasso estimator shrinks all the coordinates of the OLS estimator. When the magnitude of any coordinate of the OLS estimator is smaller than the threshold $\frac{\lambda}{2}$, lasso shrinks it to 0. As its name suggests, lasso does both shrinkage and selection and thus could produce an accurate and interpretable model in many cases.

The tuning parameter λ determines the amount of shrinkage. Larger λ's will shrink more variables to zero. A good choice of λ reflects a wise tradeoff between the accuracy of prediction and interpretation of the model.

In practice, people often want to calculate the entire solution path $\hat{\boldsymbol{\beta}}(\lambda)$, rather than a single solution for a specific value of the tuning parameter λ. This leads to algorithms that approximate the Lasso path, such as the least angle regression (LARS, Efron et al. 2004), and the coordinate descent (Friedman et al. 2008). Although the

processes for these algorithms are different, the basic ideas are the same. Starting from a large tuning parameter, which shrinks all the regression coefficients to zero, these algorithms iteratively reduce the tuning parameter and add more variables to the model. The process of solving for the regression coefficients for a new tuning parameter is based on the result of the previous tuning parameter on the solution path, so there is no need to solve the minimization problem (2.15) separately for each tuning parameter. In general, these algorithms are very effective and efficient in constructing the solution path for lasso.

The Power Family

However, lasso also has some limitations. As pointed out by Zou and Hastie (2005), when $p > n$, lasso can select at most n relevant covariates. Also, if there is a group of relevant variables that are highly correlated with each other, lasso tends to select only one of them and disregard others. Finally, as is shown by Tibshirani (1996), when $n > p$ and the input variables are highly correlated, the prediction performance of lasso is dominated by ridge regression.

Since both ridge regression and the lasso have their own benefits and drawbacks, one might want to combine these two methods so that they could fix each other's problems. One way of doing such combination is to use the *power family*

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + \lambda(\sum_{j=1}^{p} |\beta_j|^\gamma), \tag{2.16}$$

where $1 \leqslant \gamma \leqslant 2$. Friedman (2008) pointed out that when $\gamma > 1$, all coefficients of the solution are non-zero, but their dispersion increases as γ goes down to 1. He also considered the case where $0 \leqslant \gamma < 1$. When $\gamma = 0$, the problem reduces to all-subsets regression, which gives the sparsest solution. When $0 < \gamma < 1$, the minimization problem (2.14) is not convex, thus requiring much more complex solution techniques.

Elastic Net

Another way of combining ridge regression and Lasso is the *elastic net*, which was introduced by Zou and Hastie (2005). Instead of using the power family, they took the linear combination of the L_1 and L_2 penalty, so the minimization problem becomes

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\|\boldsymbol{\beta}\|_2^2. \tag{2.17}$$

Here there are two tuning parameters λ_1 and λ_2. The idea behind this combined penalty is to use the L_1 norm to do shrinkage and selection and the L_2 norm to address high correlation. Problem (2.17) can be transformed into a lasso type minimization problem. Define

$$X^* = \frac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} X \\ \sqrt{\lambda_2} I \end{pmatrix}, \quad \mathbf{y}^* = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}.$$

Let $\hat{\boldsymbol{\beta}}^{naive}$ be the solution to (2.17). Then it can be shown that

$$\hat{\boldsymbol{\beta}}^{naive} = \frac{1}{\sqrt{1+\lambda_2}} \left(\arg\min_{\boldsymbol{\beta}} \|\mathbf{y}^* - X^*\boldsymbol{\beta}\|_2^2 + \frac{\lambda_1}{\sqrt{1+\lambda_2}}\lambda_1\|\boldsymbol{\beta}\|_1 \right).$$

Therefore, the techniques for solving the Lasso optimization problem (2.15) can be applied to solve the elastic net optimization problem (2.17).

Note that $\hat{\boldsymbol{\beta}}^{naive}$ has been doubly shrunk by both the L_1 and L_2 penalty terms, so it would have high bias. The elastic net estimator is defined as

$$\hat{\boldsymbol{\beta}}^{enet} = (1+\lambda_2)\hat{\boldsymbol{\beta}}^{naive}$$

in order to fix such bias. The elastic net can be viewed as a two-stage approach: a ridge type shrinkage plus a lasso type thresholding. Zou and Hastie (2005) show that the elastic net has a "grouping effect", which means that when there is a group of variables that are highly correlated with each other, the elastic net would assign nearly equal coefficients to all of them. This clearly fixes the problem of the lasso. Zou and Hastie did simulation studies and showed that the elastic net dominates lasso

both in prediction accuracy and variable selection.

However, elastic net also has some drawbacks. First, since elastic net uses a linear combination of the ridge and lasso type penalties, the corresponding estimator is in some sense a compromise between these two methods, and if one of them does not work well, the elastic net estimator will be affected. Also, there are two tuning parameters λ_1 and λ_2 in the elastic net optimization problem (2.17). In order to find the optimal tuning parameters, we have to search within a 2D set of grid points. This adds lots of computation cost compared to ridge and lasso, which only require searching within a 1D set of parameters.

Besides the methods discussed above, there are many other regularization methods and we will not go over them in detail. Each of these methods has its own advantages and drawbacks. None of them is perfect. There is not a single regularization method that beats others in all types of problems. But in general, the regularization method is a successful group of methods that effectively solves the high-dimensional sparse linear regression problem.

2.2.2 Application of Regularization Methods to the Accelerated Failure Time Model

Wang et al. (2008) propose to implement regularization methods iteratively when some of the outputs y's in the training samples are censored. Similar to the uncensored case, they estimate $\boldsymbol{\beta}$ by solving the following minimization problem:

$$\min_{\beta} \|\mathbf{y}^* - X^*\boldsymbol{\beta}\|^2 + \lambda P(\boldsymbol{\beta}). \tag{2.18}$$

But the difference here is that $\mathbf{y}^*$ is a function of $\boldsymbol{\beta}$, so it is difficult to solve the above problem directly. Instead, $\boldsymbol{\beta}$ was solved by iteration. They start with an initial guess of $\boldsymbol{\beta}$. During each iteration, they use the current estimate of $\boldsymbol{\beta}$ to generate the Kaplan-Meier estimator $\hat{F}$ of F and use it to impute $\mathbf{y}^*$ through (2.11), then solve problem (2.18) with $\mathbf{y}^*$ substituted by the imputed one and hence update estimator of $\boldsymbol{\beta}$. This process is repeated until convergence.

The tuning parameter λ was chosen by cross validation, a very popular method for selecting the model with good prediction performance. The optimal tuning parameter λ^* is the one whose corresponding model has the smallest prediction error estimated by cross validation. We discuss the properties of cross validation in Chapter 4.

Although the above procedure sounds reasonable, it suffers from some serious problems. First, the computational cost is often very high. In most cases, the minimization problem (2.18) does not have a closed form solution, so it has to be solved numerically, which is often expensive. The iteration plus cross validation make the cost even higher. Second, there is no guarantee that the above procedure will converge. Even if it converges, it is not always true that the converged solution is the real regression coefficient. Wang et al. (2008) applied the elastic net regularization method iteratively (which they called the *Doubly Penalized Buckley James Method*) to some censored data. They showed that in several cases, the iteration will not converge. Even if the iteration converges in some other cases, the converged result is far from the real $\boldsymbol{\beta}$. In addition, because of the high computational cost, their examples are limited to moderate dimension.

In this chapter, we apply the gradient boosting machine to build the high-dimensional accelerated failure time model. The basic idea is to update $\boldsymbol{\beta}$ and the estimated distribution F alternatively during each iteration so they can benefit from each other. When the iteration terminates, the estimator $\hat{F}$ will be consistent with F, so it could be used to impute the output variable y. Then we apply an orthogonal greedy algorithm to the imputed data, which selects relevant variables and estimates the corresponding regression coefficients consistently.

2.3 Gradient Boosting Machine on the Accelerated Failure Time Model

In the high-dimensional accelerated failure time model, the problem of high dimensionality is coupled with censoring. In this section, we apply the gradient boosting machine to fit the accelerated failure time model with this high-dimensional and censored

data. As in ordinary regression problems, we use the square loss $L(y, f) = \frac{1}{2}(y - f)^2$, so the gradient boosting machine reduces to the PGA. Recall that in the Buckley James method, we need to estimate the coefficient vector $\boldsymbol{\beta}$ from the error distribution F, which has to be estimated from $\boldsymbol{\beta}$ itself. During each of the PGA iterations, we add one variable to the current model and update the corresponding coefficient. Before starting the next iteration, we add an extra step to update the estimator $\hat{F}$ of F based on the updated coefficient. Instead of performing both updates simultaneously, we propose to de-couple the two estimates, $\hat{\boldsymbol{\beta}}$ and $\hat{F}$.

The reason is that in order to get a good estimator of $\hat{F}$, we may have to compromise the model selection. De-coupling makes it possible to compute $\hat{F}$ from a model with a "larger dimension".

Here is a sketch of our de-coupling algorithm.

Algorithm 3 Two Stage Pure Greedy Algorithm on High-Dimensional Censored Data

$\hat{\boldsymbol{\beta}}^0 = (0, \ldots, 0)^T$. $\hat{f}^{(0)}(\mathbf{x}) = 0$.

for k = *1, . . . ,* m ***do***

$\hat{u}_t^{k-1} = y_t - \hat{\beta}^{(k-1)T} x_t$.

Use $\hat{u}_1^{k-1}, \ldots, \hat{u}_n^{k-1}$ *to compute a Kaplan-Meier estimator of the residual, and then impute the censored residuals. Let* $\hat{r}_1^{k-1}, \ldots, \hat{r}_n^{k-1}$ *be the imputed residuals.*

$(\hat{j}_k, \hat{\boldsymbol{\beta}}_k) = \arg\min_{j,\boldsymbol{\beta}} \frac{1}{n} \sum_{t=1}^n [\hat{r}_t^{k-1} - \beta_j x_{tj}]^2 = \arg\max_{1 \leqslant j \leqslant p_n} \frac{|\frac{1}{n} \sum_{t=1}^n \hat{r}_t^{k-1} x_{tj}|}{\sqrt{\frac{1}{n} \sum_{t=1}^n x_{tj}^2}}$.

$\hat{f}^{(k)}(\mathbf{x}) = \hat{f}^{(k-1)}(\mathbf{x}) + \hat{\beta}_k x_{\hat{j}_k} = \hat{\boldsymbol{\beta}}^{(k-1)T} x + \hat{\beta}_k x_{\hat{j}_k} = \hat{\boldsymbol{\beta}}^{(k)T} \mathbf{x}$.

end for

At each iteration, we first impute the censored residual, and then use the imputed residual to find the next variable and the corresponding coefficient. The consistency of the Kaplan-Meier estimator guarantees that the average sum of squares

$$\frac{1}{n} \sum_{t=1}^n (\hat{r}_t^{k-1} - \hat{\beta}_k x_{t\hat{j}_k})^2 \tag{2.19}$$

at each step is asymptotically equivalent to the sum of squares with uncensored

residuals. Therefore, as the iteration goes on, the imputed average of sum of squares will converge to the average of squared errors

$$\frac{1}{n}\sum_{t=1}^{n}\varepsilon_t^2. \tag{2.20}$$

We continue running the PGA until no more progress can be made in reducing the imputed average of sum of squares, which is then be very close to the mean of squared errors. In this case, the linear combination of the added variables

$$\hat{y}_t = \sum_{k=1}^{m}\hat{\beta}_k x_{t\hat{j}_k}$$

will be very close to the unobserved survival time y_t. But since we iterate too much, many irrelevant variables will be included, so the corresponding model

$$f(\mathbf{x}) = \sum_{k=1}^{m}\hat{\beta}_k x_{\hat{j}_k}$$

will be highly overfitting. That is, it will have poor prediction performance when being applied to a new set of data that is independent of the training samples. In the next section, we implement an *Orthogonal Greedy Algorithm* to rebuild the model on the covariates $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and imputed outcomes $\hat{y}_1, \ldots, \hat{y}_n$. This algorithm, coupled with a trim technique, will generate a cleaner model with irrelevant variables eliminated.

2.4 The Orthogonal Greedy Algorithm

In this section, we temporarily go back to the ordinary linear regression problem where there is no censoring. The Orthogonal Greedy Algorithm (OGA) was proposed by Ing & Lai (2011). The purpose of OGA is to improve the efficiency of the PGA iterations. We have illustrated in the previous chapter that PGA (L_2 boosting algorithm) can be applied to solve high-dimensional sparse linear regression problems. However, it turns out that PGA is not very efficient, because the same variable can be added

to the model multiple times. The *Orthogonal Greedy Algorithm* (OGA) solves this problem by orthogonizing the covariates that are sequentially added to the model. The description of OGA is as follows.

Algorithm 4 The Orthogonal Greedy Algorithm

$\tilde{\beta}^0 = (0, \ldots, 0)^T$. $\hat{f}^{(0)} = 0$.

for *k = 1, . . . , m* ***do***

$\hat{r}_t^{k-1} = y_t - \hat{f}^{k-1}(x_t)$.

$(\hat{j}_k, \tilde{\beta}_k) = \arg\min_{j,\beta} \frac{1}{n} \sum_{t=1}^n [\hat{r}_t^{k-1} - \beta x_{tj}]^2 = \arg\max_{1 \leqslant j \leqslant p_n} \frac{|\frac{1}{n} \sum_{t=1}^n \hat{r}_t^{k-1} x_{tj}|}{\sqrt{\frac{1}{n} \sum_{t=1}^n x_{tj}^2}}$.

Let $\mathbf{x}_{\hat{j}_1}^{\perp} = \mathbf{x}_{\hat{j}_1} = (x_{1\hat{j}_1}, \ldots, x_{n\hat{j}_1})^T$. *If* $k \geqslant 2$, *let* $\hat{\mathbf{x}}_{\hat{j}_k}$ *be the projection of* $\mathbf{x}_{\hat{j}_k} = (x_{1\hat{j}_k}, \ldots, x_{n\hat{j}_k})^T$ *onto the linear space spanned by* $\{\mathbf{x}_{\hat{j}_1}^{\perp}, \ldots, \mathbf{x}_{\hat{j}_{k-1}}^{\perp}\}$. *Let* $\mathbf{x}_{\hat{j}_k}^{\perp} = \mathbf{x}_{\hat{j}_k} - \hat{\mathbf{x}}_{\hat{j}_k}$.

$\hat{f}^k(x_t) = \hat{f}^{k-1}(x_t) + \tilde{\beta}_k x_{t\hat{j}_k}$.

end for

Like PGA, OGA chooses the covariate that is most correlated to the current residual during each iteration. However, before adding a new covariate to the current model, OGA does a linear transformation so that it is orthogonal to all the covariates that have already been added. This orthogonalization guarantees that each covariate would not be selected more that once. When OGA terminates, it outputs a model in terms of $\mathbf{x}_{\hat{j}_1}^{\perp}, \ldots, \mathbf{x}_{\hat{j}_m}^{\perp}$:

$$\hat{f}^m(\mathbf{x}_t) = \tilde{\beta}_1 x_{t\hat{j}_1}^{\perp} + \cdots + \tilde{\beta}_m x_{t\hat{j}_m}^{\perp}. \tag{2.21}$$

In order for the model $\hat{f}^m(x_t)$ to be used for prediction, it has to be transformed into a linear function of the original covariates

In the process of computing $\hat{\mathbf{x}}_{\hat{j}_k}$ in the OGA, let

$$\hat{\mathbf{x}}_{\hat{j}_k} = a_{k1} \hat{\mathbf{x}}_{\hat{j}_1}^{\perp} + \cdots + a_{k,k-1} \hat{\mathbf{x}}_{\hat{j}_{k-1}}^{\perp}.$$

It is easy to see that

$$a_{ki} = \frac{\sum_{t=1}^n x_{t\hat{j}_k} x_{t\hat{j}_i}^{\perp}}{\sum_{t=1}^n (x_{t\hat{j}_i}^{\perp})^2} \quad \text{for } 1 \leqslant i \leqslant k-1.$$

Suppose

$$\mathbf{x}_{\hat{j}_k}^{\perp} = \mathbf{x}_{\hat{j}_k} + b_{k1}\mathbf{x}_{\hat{j}_1} + \cdots + b_{k,k-1}\mathbf{x}_{\hat{j}_{k-1}}.$$

Then we have the relationship

$$b_{k,k-1} = -a_{k,k-1} \text{ and } b_{ki} = -a_{ki} - \sum_{l=i+1}^{k-1} a_{kl}b_{li} \text{ for } 1 \leqslant i \leqslant k-1, 2 \leqslant k \leqslant m$$

and we can rewrite (2.21) in the form

$$\hat{f}^m(\mathbf{x}_t) = \hat{\beta}_1 x_{t\hat{j}_1}^{\perp} + \cdots + \hat{\beta}_m x_{t\hat{j}_m}^{\perp} \tag{2.22}$$

with

$$\hat{\beta}_i = b_{ii}\tilde{\beta}_i + \cdots + b_{mi}\tilde{\beta}_m, \quad \text{for } i = 1, \ldots, m.$$

Ing & Lai (2011) suggest that the number of iterations m should be chosen by a *high-dimensional information criterion* (HDIC). To be specific, let J be a non-empty subset of $\{1, \ldots, p\}$, let $\hat{y}_{t;J}$ be the fitted values of applying OGA on J, and let

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{t=1}^{n}(y_t - y_{t;J})^2.$$

Define the HDIC on subset J as

$$\text{HDIC}(J) = n \log \hat{\sigma}_J^2 + \#(J)\omega_n \log p, \tag{2.23}$$

where different definitions of ω_n correspond to different criteria. For example, $\omega_n = \log n$ corresponds to HDBIC and $\omega_n = c \log\log n$ with $c > 0$ corresponds to HDHQ. The number of iterations is then chosen the integer $\hat{k}_n$ such that the HDIC (2.23) is minimized:

$$\hat{k}_n = \arg \min_{1 \leqslant k \leqslant K_n} \text{HDIC}(\hat{J}_k), \tag{2.24}$$

where K_n is a pre-specified number that is sufficiently large.

Let $\hat{J}_k = \{\hat{j}_1, \ldots, \hat{j}_k\}$. Ing & Lai (2011) show that under some sparsity condition, with probability approaching 1, $\hat{k}_n$ equals the minimum integer k such that $\hat{J}_k$ contains all relevant variables. However, $\hat{J}_{\hat{k}_n}$ may still contain irrelevant covariates. Ing & Lai (2011) then propose a back trimming technique to exclude those irrelevant covariates. For $\hat{k}_n > 1$, define

$$\hat{N}_n = \{\hat{j}_l : \mathrm{HDIC}(\hat{J}_{\hat{k}_n} - \{\hat{j}_l\}) > \mathrm{HDIC}(\hat{J}_{\hat{k}_n}), 1 \leqslant l \leqslant \hat{k}_n\}.$$

The computation of $\hat{N}_n$ only needs $\hat{k}_n - 1$ least squares estimates. Ing & Lai (2011) proved that with probability approaching 1, $\hat{N}_n$ equals the set of all relevant variables. So after the back trimming step, the model will include all relevant variables and no irrelevant variables.

2.5 PGA + OGA on Censored Data

Now we are ready to present the entire algorithm for fitting the accelerated failure time model from censored data. Our algorithm contains two stages. In the first stage, we implement PGA, which estimates the regression coefficients and imputes the censored residuals iteratively. The PGA iteration is taken until no more progress can be made in terms of the average sum of squares (3.41), which then converges to the average of squared errors (3.18). In this case, the corresponding approximation $\hat{y}_1, \ldots, \hat{y}_n$ will be close to the unobserved survival times $y_1, \ldots, y_n$. In the second stage, we apply the OGA + HDIC + Trim technique to the imputed data, just as in the uncensored case, to eliminate irrelevant variables and estimate the regression coefficient $\boldsymbol{\beta}$. This two-stage algorithm addresses the difficulties of censoring and high-dimensionality, and is able to select the correct set of relevant variables.

We now illustrate the performance of our algorithm through some numerical examples.

Example 2.1. Assume that the number of covariates $p = 1000$,
$\beta_{1-9} = (3.2, 3.2, 3.2, 3.2, 4.4, 4.4, 3.5, 3.5, 3.5)$, and $\beta_j = 0$ for $j \geqslant 10$. Let the sample

n	Censoring Rate	Number of relevant variables	Number of irrelevant variables	MSE
400	67%	8.99	1.85	3.0
400	50%	9	0.11	1.3
400	25%	9	0.007	1.1
200	67%	3.34	3.97	75.5
200	50%	7.15	2.2	26.8
200	25%	9	0.007	3.6
100	50%	0.73	2.8	160.4
100	25%	1.29	2.3	138.1
100	0%	7.524	2.8	28.3

Table 2.1: Simulation Results for Example 1

size n be 400, 200 and 100. Define covariate matrix X as

$$X_{ij} = z_{ij} + w_i, i = 1, \ldots, n, j = 1, \ldots, p,$$

where z_{ij} are i.i.d. $\mathcal{N}(1, 1)$ random variables and w_i are i.i.d. $\mathcal{N}(0, 1)$ random variables. So there are correlation among different entries of each row $\mathbf{x}_i = (X_{i1}, \ldots, X_{ip})$, but two rows $\mathbf{x}_i$ and $\mathbf{x}_j$ are independent. The noise term ε_i is defined as i.i.d. $\mathcal{N}(0, 2.25)$. The censoring time C_i is generated from a uniform distribution, and we adjust the lower and upper bounds of the distribution so that the rate of censoring is approximately 67%, 50% and 25%. We apply our two-stage algorithm to this simulated data. Then we generate a new set of test samples with size $N = 5000$ from the same distribution as the training samples, and we use the following *Mean Squared Error* to measure the prediction performance of the model fitted by our algorithm:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \sum_{j=1}^{p}\hat{\beta}_j x_{ij})^2.$$

Table 2.1 summarizes the simulation results. For $n = 100$, we could not afford the high censoring rate 67%. Instead, we adjust the censoring rate to 50%, 25% and 0%. These results are the averages of 200 simulations.

We see that the results for $n = 200$ with low censoring rate and $n = 400$ for all levels of censoring rates are good in terms of both variable selection and prediction.

	Doubly Penalized	Two stage (PGA/JB + OGA)
$\sum_{j=1}^{30} \lvert\beta_j\rvert$	89.9	69.8
$\sum_{j=31}^{60} \lvert\beta_j\rvert$	85.2	78.2
$\sum_{j=61}^{90} \lvert\beta_j\rvert$	106.7	1.2
$\sum_{j=91}^{120} \lvert\beta_j\rvert$	116.6	2.1
MSE	14188.1	1835.7

Table 2.2: Simulation Results for Example 2

However, the result is not very satisfactory when the sample size is as low as 100. This, indeed, is a hard problem, considering the high correlation among all the covariates. As reflected by the table, even if there is no censoring, we still cannot get a good result in either model selection or prediction.

Example 2.2 Let $n = 50$, $p = 120$. Define $\boldsymbol{\beta}$ as the following: β_{1-60} are i.i.d. samples generated from $\mathcal{N}(3, 1)$ distribution, and $\beta_j = 0$ for $j > 60$. The input vectors $\mathbf{x}_t$, $t = 1, \ldots, n$ are generated from a multivariate normal distribution with mean 0 and covariance matrix

$$V = \begin{pmatrix} A & & & \\ & A & J & \\ & J & A & \\ & & & A \end{pmatrix},$$

where A is a 30×30 matrix with diagonal entries 1 and other entries 0.7, and J is a 30×30 matrix with all entries 0.2. The remaining entries of V are 0. This design of the covariance matrix mimics the nature of groups of genes. In practice, there could be high correlation among genes in the same group, while low or no correlation among genes in different groups. As in the previous example, the censoring time C_i is generated from a uniform distribution. We adjust the lower and upper bound of the distribution so that the rate of censoring is around 50%. The noise ε_i are i.i.d. $\mathcal{N}(0, 1)$ random variables.

Here we compare the result of our two-stage algorithm with the doubly penalized

Buckley-James method (the iterative elastic net method) introduced by Wang et al. (2008). We only report the summation of absolute values of the coefficients in each group and the MSE. The result can be seen in Table 2.5. Again, the values are averages of 200 simulations.

We see that the regularization method failed to distinguish irrelevant variables from relevant variables. In contrast, our two-stage algorithm was able to select relevant variables and eliminate irrelevant variables. Because of this, it obtains a much better MSE than the regularization method.

Chapter 3

Asymptotic Theory of the Gradient Boosting Machine

In the previous chapters we reviewed the gradient boosting machine as an algorithm for fitting additive models, and implemented it on a special case where the nonlinear parameter is infinite-dimensional. In this chapter, we formally prove the convergence and consistency of the gradient boosting machine. At the beginning, we go over some regularity conditions and propose the main theory, which claims that as the sample size n and number of iterations m go to infinity, the model fitted by the gradient boosting machine converges to the real model in the L_2 norm. We start proving the main theorem by showing the L_2 convergence of an "ideal" version of the gradient boosting machine, which works on the joint distribution of the input $\mathbf{X}$ and output Y. Then we use large deviation theory to bound the difference between the the "practical" algorithm and the "ideal" algorithm. It turns out that the nonlinear parameters r_k add much difficulty to establish such bound. We overcome this difficulty by dividing the set of basis functions into finite segments and controlling the difference within each segment. This result will allow us to view the whole set as finite and make it possible to control the difference between the outcomes of the "practical" and "ideal" algorithms. Then we establish the asymptotic theory for the "practical" gradient boosting machine based on these controlled bounds. We close this chapter with some

discussion on the stopping criterion of the algorithm.

3.1 General Settings of the Gradient Boosting Machine

As discussed in the previous chapters, we assume that model $f(\cdot)$ is a linear combination of basis functions. That is,

$$f(\mathbf{x}) = \sum_{k=1}^{N} \beta_{j_k} \phi(x_{j_k}; r_k).$$

Let Γ be the space of nonlinear parameter r_j's. At the beginning, we only consider one-dimensional parameters. We will show later that our result can be easily generalized to muti-dimensional parameters. So here we assume that $\Gamma \subset \mathbb{R}$. Let the output variable Y be

$$Y = f(\mathbf{X}) + \varepsilon,$$

where ε is independent of f, has mean 0 and finite variance. Let $L(\cdot, \cdot) \geqslant 0$ be the loss function. We assume that L is strictly convex in the second argument and $L(y, f) = 0$ if and only if $y = f$. The real regression function $f(\cdot)$ then minimizes the expected value of the loss function, i.e.

$$f = \arg \min_{g \text{ is a linear combination of the basis functions } \phi(X_{j_k}; r_k)} \mathbf{E} L(Y, g(\mathbf{X})).$$

where the expectation is taken on the joint distribution of $\mathbf{X}$ and Y.

We assume that the number of covariates p is in the order of an exponential function of n:

$$p_n = \exp(O(n^{\xi})) \quad \text{for some } 0 < \xi < 1. \tag{3.1}$$

Here we use the notation p_n to indicate that p could increase with n. We also assume that

$$N = O(p_n). \tag{3.2}$$

Although we allow p to be much larger than n, the real model is limited to be sparse. That is, only a few covariates (basis functions) are involved in the additive model. To be specific, we require the following sparsity condition.

$$\sum_{k=1}^{N} |\beta_{j_k}| \leqslant M, \tag{3.3}$$

where M is a positive constant that does not change with n (or N). We also impose some regularity conditions on the basis functions

$$m_1 \leqslant \sqrt{\mathbf{E}\phi(x_j; r)^2} \leqslant M_1, j = 1, \ldots, p_n, r \in \Gamma. \tag{3.4}$$

$$\mathbf{E}|\phi(x_j; r)| \leqslant M_2, j = 1, \ldots, p_n, r \in \Gamma. \tag{3.5}$$

$$\mathbf{E}\phi(x_j; r)^4 < +\infty, j = 1, \ldots, p_n, r \in \Gamma. \tag{3.6}$$

$$\Gamma \subset [\gamma_1, \gamma_2],\ \gamma_1, \gamma_2 \in \mathbb{R},\ \gamma_2 - \gamma_1 = M_3. \tag{3.7}$$

These conditions are very general. Most models used in practice will satisfy these regularity conditions.

For each covariate X_j, define $\phi_{x_j}(r) = \phi(x_j; r)$. We assume that $\phi_{x_j}(r)$ has continuous derivative in r and

$$|\phi'_{x_j}(r)| \leqslant M_4, j = 1, \ldots, p_n, r \in \Gamma. \tag{3.8}$$

To study the consistency of the gradient boosting algorithm, we also assume that the loss function $L(\cdot, \cdot)$ is twice continuous differentiable in the second argument, and $\exists c_1, c_2 > 0$, such that

$$c_1 \leqslant \frac{\partial^2 L}{\partial g^2}(y, g) \leqslant c_2, \forall y, g. \tag{3.9}$$

Given all the assumptions above, we are able to propose the main theory, which

specfies the rate of convergence of the gradient boosting algorithm with moderate number of iterations.

Theorem 1 *Let $m = m_n = o(\log n) \to +\infty$. Then under assumptions* (3.1) *to* (3.9)*,*

$$\mathbf{E}(\hat{f}^m(\mathbf{X}) - f(\mathbf{X}))^2 = O((1 - \frac{c_1}{c_2})^{\frac{m}{2}}),$$

where $\hat{f}(\cdot)$ is the model fitted at the m-th iteration of the gradient boosting machine, and the expectation is taken in the distribution of the random vector $\mathbf{X}$.

3.2 Population Versions of the Gradient Boosting Machine

Recall that during each iteration of the gradient boosting machine, we first select the basis function that is most correlated with the current descent direction $\hat{u}^{k-1}$

$$(\hat{j}_k, \hat{b}_k) = \arg \max_{1 \leqslant j \leqslant p, b \in \mathbb{R}} \frac{|\frac{1}{n} \sum_{t=1}^n \hat{u}_t^{k-1} \phi(x_{tj}; b)|}{\sqrt{\frac{1}{n} \sum_{t=1}^n \phi(x_{tj}; b)^2}},$$

and then choose the corresponding coefficient by minimizing the expected loss function

$$\hat{\rho}_k = \arg \min_{\rho} \frac{1}{n} \sum_{t=1}^n L(y_t, \hat{f}^{k-1}(\mathbf{x}_t) + \rho \phi(x_{t\hat{j}_k}; \hat{b}_k)).$$

Here the "correlation" and "expectation" are estimated by taking the average on the training samples. The reason is that in practice, we only have these finite number of training samples, and the best we can do is to use them to approximate the real correlation and expectation. Ideally, if we know the joint distribution of $\mathbf{X}$ and Y, we will be able to evaluate the "real" correlation and expectation instead of using the sample average, then the algorithm should be more accurate. So what we are really trying to implement is the following *population version* of the gradient boosting machine

Algorithm 5 The Population Version of the Gradient Boosting Machine

$f^0 = 0$.

for k = *1, . . . ,* m ***do***

$u^{k-1}(\mathbf{x}) = -\frac{\partial L}{\partial f}(y, f^{k-1}(\mathbf{x}))$.

$(j_k, b_k) = \arg\max_{1 \leqslant j \leqslant p_n, b \in \Gamma} (\mathbf{E}\phi(X_j; b)^2)^{-\frac{1}{2}} |\mathbf{E}u^{k-1}(\mathbf{X})\phi(X_j; b)|$.

$\rho_k = \arg\min_\rho \mathbf{E}L(Y, f^{k-1}(\mathbf{X}) + \rho\phi(X_{j_k}; b_k))$.

$f^k(\mathbf{x}) = f^{k-1}(\mathbf{x}) + \rho_k\phi(x_{j_k}; b_k)$.

end for

Let's emphasize that the above algorithm is just an ideal version, which is not feasible in practice. Here we need it in order to study the convergence and consistency of the gradient boosting machine.

Algorithm 5 insists that during each iteration we need to find the very best basis function that is most correlated with the current negative gradient. This criterion is too demanding. In our analysis, a weaker condition is more desirable. Instead of finding the "optimal" basis function, we can use one of the "suboptimal" basis functions that is within some range of the "optimal" one. So we have the following weak version of the gradient boosting machine.

Algorithm 6 The Weak Gradient Boosting Machine

$f^0 = 0$.

for k = *1, . . . ,* m ***do***

$u^{k-1}(\mathbf{x}) = -\frac{\partial L}{\partial f}(y, f^{k-1}(\mathbf{x}))$.

j_k, b_k *is such that*

$(\mathbf{E}\phi(X_{j_k}; b_k)^2)^{-\frac{1}{2}} |\mathbf{E}u^{k-1}(\mathbf{X})\phi(X_{j_k}; b_k)| \geqslant \tau \max_{1 \leqslant j \leqslant n, b \in \Gamma} (\mathbf{E}\phi(X_j; b)^2)^{-\frac{1}{2}} |\mathbf{E}u^{k-1}(\mathbf{X})\phi(X_j; b)|$,

for some $0 < \tau < 1$.

$\rho_k = \arg\min_\rho \mathbf{E}L(y, f^{k-1}(\mathbf{X}) + \rho\phi(X_{j_k}; b_k))$.

$f^k(\mathbf{x}) = f^{k-1}(\mathbf{x}) + \rho_k\phi(x_{j_k}; b_k)$.

end for

In order to be distinguished from the population version, the original algorithm (Algorithm 2) is also called the *sample version* of the gradient boosting machine.

During each iteration, the basis function chosen by the population algorithms is highly correlated to the current steepest descent direction. This guarantees that the expected loss is reduced iteratively and the fitted model f^k converges to the real model f. This result is shown in the following theorem.

Theorem 2 *Suppose* $\{f^m\}$ *the outcome of the m-th iteration of the weak gradient boosting machine (Algorithm* 6*). Then*

$$\mathbf{E}(f^m(\mathbf{X}) - f(\mathbf{X}))^2 = O((1 - \frac{c_1}{c_2})^{\frac{m}{2}}).$$

Proof For simplicity, we use f^k to represent the random variable $f^k(\mathbf{X})$ and u^k to represent the random variable $u^k(\mathbf{X})$. We have

$$\begin{aligned}
\mathbf{E}L(Y, f^k) &= \mathbf{E}L(Y, f^{k-1} + \rho_k \phi(X_{j_k}; b_k)) \\
&= \mathbf{E}L(Y, f^{k-1}) + \rho_k \mathbf{E}\frac{\partial L}{\partial f}(Y, f^{k-1})\phi(X_{j_k}; b_k) + \frac{\rho_k^2}{2}\mathbf{E}\frac{\partial^2 L}{\partial f^2}(Y, \eta)\phi^2(X_{j_k}; b_k) \\
&\leqslant \mathbf{E}L(Y, f^{k-1}) - \rho_k \mathbf{E}u^{k-1}\phi(X_{j_k}; b_k) + \frac{\rho_k^2 c_2}{2}\mathbf{E}\phi^2(X_{j_k}; b_k).
\end{aligned}$$

Here η is a random variable that is between f^k and f^{k-1}. The right hand side of the above inequality is quadratic in ρ_k, whose minimal value is $\mathbf{E}L(Y, f^{k-1}) - \frac{(\mathbf{E}u^{k-1}\phi(X_{j_k};b_k))^2}{2c_2\mathbf{E}\phi^2(X_{j_k};b_k)}$. Therefore

$$\mathbf{E}L(Y, f^k) \leqslant \mathbf{E}L(Y, f^{k-1}) - \frac{(\mathbf{E}u^{k-1}\phi(X_{j_k}; b_k))^2}{2c_2\mathbf{E}\phi^2(X_{j_k}; b_k)}. \tag{3.10}$$

Similarly, we have

$$\mathbf{E}L(Y, f^k) \geqslant \mathbf{E}L(Y, f^{k-1}) - \rho_k \mathbf{E}u^{k-1}\phi(X_{j_k}; b_k) + \frac{\rho_k^2 c_1}{2}\mathbf{E}\phi^2(X_{j_k}; b_k).$$

But

$$\mathbf{E}L(Y, f^k) \leqslant \mathbf{E}L(Y, f^{k-1}),$$

so we have

$$\rho_k \mathbf{E} u^{k-1} \phi(X_{j_k}; b_k) \geqslant \frac{\rho_k^2 c_1}{2} \mathbf{E} \phi^2(X_{j_k}; b_k).$$

Hence

$$|\rho_k| \leqslant \frac{2|\mathbf{E} u^{k-1} \phi(X_{j_k}; b_k)|}{c_1 \mathbf{E} \phi^2(X_{j_k}; b_k)}. \tag{3.11}$$

We could also have

$$\begin{aligned} \mathbf{E} L(Y, f^k) &\leqslant \mathbf{E} L(Y, f^{k-1}) - \rho_k \mathbf{E} u^{k-1} \phi(X_{j_k}; b_k) + \frac{\rho_k^2 c_2}{2} \mathbf{E} \phi^2(X_{j_k}; b_k) \\ &\leqslant \mathbf{E} L(Y, f^{k-1}) + \rho_k (\mathbf{E}(u^{k-1})^2 \mathbf{E} \phi^2(X_{j_k}; b_k))^{\frac{1}{2}} + \frac{\rho_k^2 c_2}{2} \mathbf{E} \phi^2(X_{j_k}; b_k). \end{aligned}$$

Again, taking the minimal value of the right hand side as a quadratic function of ρ_k, we get

$$\mathbf{E} L(Y, f^k) \leqslant \mathbf{E} L(Y, f^{k-1}) - \frac{\mathbf{E}(u^{k-1})^2}{2c_2}. \tag{3.12}$$

On the other hand, we have

$$\begin{aligned} L(Y, f) &= L(Y, f^{k-1}) + \frac{\partial L}{\partial f}(Y, f^{k-1})(f - f^{k-1}) + \frac{\partial^2 L}{\partial f^2}(Y, \eta) \frac{(f - f^{k-1})^2}{2} \\ &\geqslant L(Y, f^{k-1}) + \frac{\partial L}{\partial f}(Y, f^{k-1})(f - f^{k-1}) + \frac{c_1}{2}(f - f^{k-1})^2 \\ &= L(Y, f^{k-1}) - u^{k-1}(f - f^{k-1}) + \frac{c_1}{2}(f - f^{k-1})^2, \end{aligned}$$

where f is the real model and η is between f and f^{k-1}. If we treat the right hand side of the inequality as a quadratic function of $f - f^{k-1}$, then it is minimized when $f - f^{k-1} = \frac{u^{k-1}}{c_1}$ with minimal value $L(Y, f^{k-1}) - \frac{(u^{k-1})^2}{2c_1}$. Therefore

$$L(Y, f) \geqslant L(Y, f^{k-1}) - \frac{(u^{k-1})^2}{2c_1}.$$

Taking expectation on both sides, and reorganizing terms, we have

$$\mathbf{E}(u^{k-1})^2 \geqslant 2c_1(\mathbf{E} L(Y, f^{k-1}) - \mathbf{E}(Y, f)). \tag{3.13}$$

Combining (3.12) and (3.13), we have

$$\begin{aligned}\mathbf{E}L(Y,f^k)-\mathbf{E}L(Y,f) &\leqslant \mathbf{E}L(Y,f^{k-1})-\mathbf{E}L(Y,f)-\frac{\mathbf{E}(u^{k-1})^2}{2c_2}\\ &\leqslant \mathbf{E}L(Y,f^{k-1})-\mathbf{E}L(Y,f)-\frac{c_1}{c_2}(\mathbf{E}L(Y,f^{k-1})-\mathbf{E}L(Y,f))\\ &= (1-\frac{c_1}{c_2})(\mathbf{E}L(Y,f^{k-1})-\mathbf{E}L(Y,f)).\end{aligned}$$

Hence we have

$$\mathbf{E}L(Y,f^k)-\mathbf{E}L(Y,f)=(1-\frac{c_1}{c_2})^k(\mathbf{E}L(Y,f^0)-\mathbf{E}L(Y,f))=O((1-\frac{c_1}{c_2})^k). \quad (3.14)$$

Using (3.10), we have

$$\frac{(\mathbf{E}u^{k-1}\phi(X_{j_k};b_k))^2}{2c_2\mathbf{E}\phi^2(X_{j_k};b_k)}\leqslant \mathbf{E}L(Y,f^{k-1})-\mathbf{E}L(Y,f^k)\leqslant \mathbf{E}L(Y,f^{k-1})-\mathbf{E}L(Y,f)=O((1-\frac{c_1}{c_2})^{k-1}). \quad (3.15)$$

According to (3.11),

$$\begin{aligned}|\rho_k| &\leqslant \frac{2|\mathbf{E}u^{k-1}\phi(X_{j_k};b_k)|}{c_1\mathbf{E}\phi^2(X_{j_k};b_k)}\\ &\leqslant \frac{2|\mathbf{E}u^{k-1}\phi(X_{j_k};b_k)|}{c_1(\mathbf{E}\phi^2(X_{j_k};b_k))^{\frac{1}{2}}}\cdot\frac{1}{(\mathbf{E}\phi^2(X_{j_k};b_k))^{\frac{1}{2}}}\\ &= O(\frac{\sqrt{c_2}}{c_1m_1}(1-\frac{c_1}{c_2})^{\frac{k-1}{2}}).\end{aligned}$$

Hence

$$\sum_{k=0}^{+\infty}|\rho_k|<+\infty.$$

This result shows that regardless of how the basis functions are chosen during each iteration, the corresponding step size ρ_k will always converge to zero as the iteration goes on. In particular, the step sizes selected by the gradient boosting machine will remain bounded. This conclusion will be used later in this chapter when we establish the asymptotic theory for the sample version gradient boosting machine. Without

loss of generality we can assume that

$$\sum_{k=0}^{+\infty} |\rho_k| < M, \tag{3.16}$$

where M is defined in (3.3).

According to the definition of the weak greedy algorithm,

$$\max_{1\leqslant j\leqslant p, b\in\Gamma} (\mathbf{E}\phi^2(X_j;b))^{-\frac{1}{2}} |\mathbf{E}u^{k-1}\phi(X_j;b)| \leqslant \frac{1}{\tau}(\mathbf{E}\phi^2(X_{j_k};b_k))^{-\frac{1}{2}} |\mathbf{E}u^{k-1}\phi(X_{j_k};b_k)| = O(\frac{\sqrt{c_2}}{\tau}(1-\frac{c_1}{c_2})^{\frac{k-1}{2}})$$

Therefore

$$\max_{1\leqslant j\leqslant p, b\in\Gamma} |\mathbf{E}u^{k-1}\phi(X_j;b)| = O(\frac{\sqrt{c_2}M_1}{\tau}(1-\frac{c_1}{c_2})^{\frac{k-1}{2}}).$$

Note that f is the real model. We have

$$\begin{aligned} \mathbf{E}L(Y,f^{k-1}) &\geqslant \mathbf{E}L(Y,f) = \mathbf{E}L(Y,f^{k-1}) - \mathbf{E}u^{k-1}(f-f^{k-1}) + \mathbf{E}\frac{\partial^2 L}{\partial f^2}(Y,\eta)(f-f^{k-1})^2 \\ &\geqslant \mathbf{E}L(Y,f^{k-1}) - \mathbf{E}u^{k-1}(f-f^{k-1}) + c_1\mathbf{E}(f-f^{k-1})^2. \end{aligned}$$

Therefore

$$\mathbf{E}u^{k-1}(f-f^{k-1}) \geqslant c_1\mathbf{E}(f-f^{k-1})^2. \tag{3.17}$$

Therefore

$$\begin{aligned} \mathbf{E}(f-f^{k-1})^2 &\leqslant \frac{\mathbf{E}u^{k-1}(f-f^{k-1})}{c_1} \\ &\leqslant \frac{1}{c_1}\sum_{l=1}^{N} |\beta_l \mathbf{E}u^{k-1}\phi(X_l;r_l)| + \frac{1}{c_1}\sum_{i=0}^{k-1} |\rho_i \mathbf{E}u^{k-1}\phi(X_{j_i};b_i)| \\ &\leqslant \frac{2M}{c_1}\max_{1\leqslant j\leqslant p, r\in\Gamma} |\mathbf{E}u^{k-1}\phi(X_j;r)| \\ &= O(\frac{\sqrt{c_2}MM_1}{c_1\tau}(1-\frac{c_1}{c_2})^{\frac{k-1}{2}}). \end{aligned}$$

3.3 Convergence and Consistency of the Sample Version

Given the asymptotic of the population version, we can proceed to study the convergence and consistency of the sample version gradient boosting algorithm, which is the version implemented in practice. In order to use the theoretical result in the previous section, we need a way to connect the population and the sample versions. Motivated by this, we define the *semi-population version* of the gradient boosting machine:

Algorithm 7 The Semi-Population Version of the Gradient Boosting Algorithm

$\tilde{f}^0 = 0$.
Let $\hat{j}_k$, $\hat{b}_k$ *be defined as in the sample version.*
for $k = 1, \ldots, m$ ***do***
$\tilde{\rho}_k = \arg\min_\rho \mathbf{E}L(Y, \tilde{f}^{k-1}(\mathbf{X}) + \rho\phi(X_{\hat{j}_k}; \hat{b}_k))$.
$\tilde{f}^k(\mathbf{x}) = \tilde{f}^{k-1}(\mathbf{x}) + \tilde{\rho}_k\phi(x_{\hat{j}_k}; \hat{b}_k)$.
end for

Similar to the population version, the semi-population version of the gradient boosting algorithm is also implemented on joint distribution of $\mathbf{X}$ and Y. But the difference is that during each iteration, the semi-population version adds the basis function chosen by the sample version. In other words, the basis functions are selected by maximizing the sample correlation, instead of the real correlation to the descent direction. So they might not be the optimal choices. But we will show that these basis functions are "sub-optimal", i.e. the sample population version is a weak version of gradient boosting machine. Hence the asymptotic properties of the semi-population version can be verified by Theorem 2. Then we can establish the convergence and consistency of the sample version by bounding its difference with the semi-population version.

For $k = 1, \ldots, m$, let $\mathbf{w}_k$ be a sequence of tuples $((l_1, s_1), \ldots, (l_k, s_k)) \in (\{1, \ldots, p\} \times \Gamma)^k$. Define $\tilde{f}_{\mathbf{w}_0} = 0$ and

$$\tilde{f}_{\mathbf{w}_k} = \tilde{f}_{\mathbf{w}_{k-1}} + \tilde{\rho}_k \phi(X_{l_k}; s_k), \tag{3.18}$$

where

$$\tilde{\rho}_k = \arg\min_{\rho} \mathbf{E} L(Y, \tilde{f}_{\mathbf{w}_{k-1}} + \rho \phi(X_{l_k}; s_k)). \tag{3.19}$$

Similarly, define $\hat{f}_{t,\mathbf{w}_0} = 0$ for $t = 1, \ldots, n$ and

$$\hat{f}_{t,\mathbf{w}_k} = \hat{f}_{t,\mathbf{w}_{k-1}} + \hat{\rho}_k \phi(x_{tl_k}; s_k), \tag{3.20}$$

where

$$\hat{\rho}_k = \arg\min_{\rho} \frac{1}{n} \sum_{t=1}^{n} L(y_t, \hat{f}_{t,\mathbf{w}_{k-1}} + \rho \phi(x_{tl_k}; s_k)). \tag{3.21}$$

Here keep in mind that $\tilde{f}_{\mathbf{w}_k} = \tilde{f}_{\mathbf{w}_k}(\mathbf{X})$ is a function of the random vector $\mathbf{X}$, while $\hat{f}_{t,\mathbf{w}_k}$ is a fixed value defined iteratively by the finite samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$. Let $\tilde{h}_{\mathbf{w}_k} = f - \tilde{f}_{\mathbf{w}_k}$, $\hat{h}_{t,\mathbf{w}_k} = f_t - \hat{f}_{t,\mathbf{w}_k}$, $\tilde{u}_{\mathbf{w}_k} = -\frac{\partial L}{\partial f}(y, \tilde{f}_{\mathbf{w}_k})$ and $\hat{u}_{\mathbf{w}_k} = -\frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k})$. For $1 \leqslant j \leqslant p$ and $r \in \Gamma$, define

$$\tilde{\mu}_{\mathbf{w}_k,j,r} = (\mathbf{E}\phi^2(\mathbf{X}_j; r))^{-\frac{1}{2}} \mathbf{E}(\tilde{u}_{\mathbf{w}_k} \phi(\mathbf{X}_j; r))$$

and

$$\hat{\mu}_{\mathbf{w}_k,j,r} = (\frac{1}{n} \sum_{t=1}^{n} \phi^2(x_{tj}; r))^{-\frac{1}{2}} (\frac{1}{n} \sum_{t=1}^{n} \hat{u}_{t,\mathbf{w}_k} \phi(x_{tj}; r)).$$

Note that $\tilde{\mu}$ and $\hat{\mu}$ are the population and sample correlations of the steepest descent direction and basis functions, if we can show that for each basis function, the two correlations are close to each other, then intuitively the population and sample version algorithms should select the same (or very similar) basis function during each iteration, and the convergence and consistency of the population version of the algorithm (Theorem 2) should imply the same properties of its sample counterpart.

To be specific, let $0 < \delta_1 < \frac{1-\xi}{2}$, where ξ is defined in (3.1). Let $\theta \in (0, \frac{1}{3})$ and $\xi_1 \in (0,1)$, such that $0 < \theta \leqslant \frac{(c_1^2/c_2^2)\xi_1}{2+(c_1^2/c_2^2)\xi 1}$. Define $\xi_2 = \frac{2}{1-\xi_1}$. Let

$$A_n = \{ \max_{\mathbf{w}_k \in (\{1,\ldots,p\}\times\Gamma)^k, 0\leqslant k\leqslant m-1, 1\leqslant j\leqslant p, r\in\Gamma} |\hat{\mu}_{\mathbf{w}_k,j,r} - \tilde{\mu}_{\mathbf{w}_k,j,r}| \leqslant n^{-\delta_1} \}.$$

We want to bound the probability of

$$A_n^c = \{ \max_{\mathbf{w}_k \in (\{1,\ldots,p\}\times\Gamma)^k, 0\leqslant k\leqslant m-1, 1\leqslant j\leqslant p, r\in\Gamma} |\hat{\mu}_{\mathbf{w}_k,j,r} - \tilde{\mu}_{\mathbf{w}_k,j,r}| > n^{-\delta_1} \}.$$

In the next section, we will show that the absolute difference for each fixed $\mathbf{w}_k$, j and r is bounded. However, the nonlinear parameter $r \in \Gamma$ results in infinitely many basis functions in the dictionary. So the left hand side of the inequality in the definition of A_n^c is actually the maximum of infinitely many terms. This adds difficulty to estimate the upper bound of $\mathbf{P}(A_n^c)$. Our solution here is to show that the set of basis function is redundant, i.e. if $r_1 \approx r_2$, then the performance of $\phi(x_j; r_1)$ should be very close to that of $\phi(x_j; r_2)$. Based on this argument, we can divide the set Γ of parameters into a finite number of intervals, where the size of each interval is small enough so that the difference between basis functions with nonlinear parameters in the same interval can be controlled. Then $P(A_n^c)$ can be bounded by the taking the sum of the probabilities of large deviation of each covariate and each interval, which would be only finitely many terms. So the upper bound of $P(A_n^c)$ can be estimated. Actually, we can show that

$$\mathbf{P}(\tilde{A}_n^c) = \exp(-Cn^\eta), \tag{3.22}$$

for some $C > 0$, $\eta > 0$. We give the details of the proof in the next section.

Define

$$B_n = \{ \min_{0\leqslant k\leqslant m-1} \max_{1\leqslant j\leqslant p, r\in\Gamma} |\tilde{\mu}_{\hat{\mathbf{v}}_k,j,r}| > \xi_2 n^{-\delta_1} \},$$

where $\hat{\mathbf{v}}_k = ((\hat{j}_1, \hat{b}_1), \ldots, (\hat{j}_k, \hat{b}_k))$ is the sequence of parameters selected by the sample version (Algorithm 2). Then on $A_n \bigcap B_n$, we have

$$\begin{aligned} |\tilde{\mu}_{\hat{\mathbf{v}}_{k-1},\hat{j}_k,\hat{b}_k}| &\geqslant |\hat{\mu}_{\hat{\mathbf{v}}_{k-1},\hat{j}_k,\hat{b}_k}| - |\hat{\mu}_{\hat{\mathbf{v}}_{k-1},\hat{j}_k,\hat{b}_k} - \tilde{\mu}_{\hat{\mathbf{v}}_{k-1},\hat{j}_k,\hat{b}_k}| \\ &\geqslant \max_{1\leqslant j\leqslant p, r\in\Gamma} |\hat{\mu}_{\hat{\mathbf{v}}_{k-1},j,r}| - n^{-\delta_1} \\ &\geqslant \max_{1\leqslant j\leqslant p_n} |\tilde{\mu}_{\hat{\mathbf{v}}_{k-1},j,r}| - 2n^{-\delta_1} \\ &\geqslant (1-\frac{2}{\xi_2}) \max_{1\leqslant j\leqslant p,j,r} |\tilde{\mu}_{\hat{\mathbf{v}}_{k-1},j,r}| \\ &= \xi_1 \max_{1\leqslant j\leqslant p,j,r} |\tilde{\mu}_{\hat{\mathbf{v}}_{k-1},j,r}|, \end{aligned}$$

where the second inequality follows from the definition of $\hat{j}_k$, $\hat{b}_k$, and the last inequality follows from the definition of the set B_n. Hence the semi-population is a weak version of gradient boosting machine on $A_n \bigcap B_n$. According to Theorem 2, we have

$$\mathbf{E}(\tilde{f}_{\hat{\mathbf{v}}_m}(\mathbf{X}) - f(\mathbf{X}))^2 = O((1-\frac{c_1}{c_2})^{\frac{m-1}{2}}) \text{ on } A_n \bigcap B_n. \tag{3.23}$$

On B_n^c, there exists $k \leqslant m-1$, such that

$$\max_{1\leqslant j\leqslant p, r\in\Gamma} |\tilde{\mu}_{\hat{\mathbf{v}}_k,j,r}| \leqslant \xi_2 n^{-\delta_1}.$$

According to the proof of Theorem 2, we have

$$\mathbf{E}(\tilde{f}_{\hat{\mathbf{v}}_m}(\mathbf{X})-f(\mathbf{X}))^2 \leqslant \mathbf{E}(\tilde{f}_{\hat{\mathbf{v}}_k}(\mathbf{X})-f(\mathbf{X}))^2 \leqslant \frac{2M}{c_1} \max_{1\leqslant j\leqslant p, r\in\Gamma} |\mathbf{E}(\tilde{u}_{\hat{\mathbf{v}}_k}\phi(X_j;r))| \leqslant \frac{2M\xi_2}{c_1 m_1} n^{-\delta_1}. \tag{3.24}$$

Combining (3.22), (3.23) and (3.24), we have

$$\mathbf{E}(\tilde{f}_{\hat{\mathbf{v}}_m}(\mathbf{X}) - f(\mathbf{X}))^2 = O((1-\frac{c_1}{c_2})^{\frac{m}{2}}) \text{ for } m = o(\log(n)). \tag{3.25}$$

In the next section, we will also prove that

$$\mathbf{E}(\hat{f}_{\hat{\mathbf{v}}_m}(\mathbf{X}) - \tilde{f}_{\hat{\mathbf{v}}_m}(\mathbf{X}))^2 = O(n^{-2\delta_1}). \tag{3.26}$$

Combining (3.25) and (3.26), and use the fact that $m = o(\log(n))$, we have

$$\mathbf{E}(\hat{f}_{\hat{\mathbf{v}}_m}(\mathbf{X}) - f(\mathbf{X}))^2 = O((1 - \frac{c_1}{c_2})^{\frac{m}{2}}).$$

Hence Theorem 1 is proved.

3.4 Proof of (3.22) and (3.26)

Proof of (3.26)

As mentioned in the previous section, the main difficulty for bounding the probability on the left hand side of (3.22) is the infinity of the set of basis functions. We address this difficulty by dividing the set of nonlinear parameters Γ into finite subintervals. For each subinterval, we choose a single value a in it, which we call the *grid point*. In order to prove (3.22), we first establish the boundedness of the probability for the large deviation of some sample means and the corresponding expected values where the nonlinear parameter is fixed at grid points. Then we show that the difference between two basis functions whose nonlinear parameters are in the same subintervals can be controlled. Hence, we can bound the maximum probability of the large deviation for each variable and each subinterval of nonlinear parameters. Therefore, the original infinite set of basis functions can be treated as a finite set, and the problem will be greatly simplified.

For arbitrary sequences $\mathbf{l}_k = (l_1, \ldots, l_k)$, $\mathbf{s}_k = (s_1, \ldots, s_k)$ and $\boldsymbol{\rho}_k = (\rho_1, \ldots, \rho_k)$, define

$$f_{\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k} = f_{\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k}(\mathbf{X}) = \sum_{j=1}^{k} \rho_j \phi(X_{l_j}; s_j).$$

Similarily, define

$$f_{t,\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k} = \sum_{j=1}^{k} \rho_j \phi(x_{tl_j}; s_j).$$

That is, $f_{t,\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k}$ is the value of the function $f_{\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k}(\cdot)$ at the t-th sample $\mathbf{x}_t$. Using the same argument in the proof of (3.16), we can show that the step length $\hat{\rho}_k$'s for

the sample version gradient boosting machine (Algorithm 2) also satisfy

$$\sum_{k=0}^{+\infty} |\hat{\rho}_k| < M.$$

Here we just need to replace the expectation by the sample mean, and the real model f by the minimizer

$$\bar{f}_n = \arg\min_f \frac{1}{n} \sum_{t=1}^{n} L(y_t, f(\mathbf{x}_t)).$$

The argument will be the same as the proof of (3.16). For this reason, we add an extra condition

$$\sum_{j=1}^{k} |\rho_j| < M, \quad \text{as } k \to +\infty \tag{3.27}$$

to the definition of $f_{\mathbf{l}_k, \mathbf{s}_k, \boldsymbol{\rho}_k}$ and $f_{t, \mathbf{l}_k, \mathbf{s}_k, \boldsymbol{\rho}_k}$.

Recall that Γ is the set of nonlinear parameters, and $\Gamma \in [\gamma_1, \gamma_2]$. We divide the interval $[\gamma_1, \gamma_2]$ into n subintervals. To be specific, let $a^{(1)} = \gamma_1$, for $i = 1, \ldots, n$, let

$$a^{(i+1)} = \gamma_1 + \frac{i}{n} M_3 = \gamma_1 + \frac{i}{n}(\gamma_2 - \gamma_1).$$

Let $I(a^{(j)})$ be the interval $[a^{(j)}, a^{(j+1)})$ for $j = 1, \ldots, n$ and let $I(a^{(n+1)}) = \{\gamma_2\}$. $a^{(1)}, \ldots, a^{(n+1)}$ are called grid points on Γ. Let $\mathcal{A}$ be the set of all grid points. Let

$\mathbf{a}_k = (a_1, \ldots, a_k)$, where $a_j \in \mathcal{A}$, $j = 1, \ldots, k$. For $j \in \{1, \ldots, p\}$, $a \in \mathcal{A}$, define

$$\begin{aligned}\tilde{d}(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) = \max\{ & \max_{1\leqslant i\leqslant k} |\frac{1}{n}\sum_{t=1}^{n} \frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})\phi(x_{tl_i}, a_i) - \mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})\phi(X_{l_i}; a_i)|, \\ & |\frac{1}{n}\sum_{t=1}^{n} \frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_k,\mathbf{a}_k,\boldsymbol{\rho}_k})\phi(x_{tj}; a) - \mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_k,\mathbf{a}_k,\boldsymbol{\rho}_k})\phi(X_j; a)|, \\ & \max_{1\leqslant i\leqslant k} |\frac{1}{n}\sum_{t=1}^{n} \phi^2(x_{tl_i}; a_i) - \mathbf{E}\phi^2(X_{l_i}; a_i)|, \\ & \max_{1\leqslant i\leqslant k} |\frac{1}{n}\sum_{t=1}^{n} |\phi(x_{tl_i}; a_i)| - \mathbf{E}|\phi(X_{l_i}; a_i)||, \\ & \max_{1\leqslant i\leqslant k} |\frac{1}{n}\sum_{t=1}^{n} |\frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})| - \mathbf{E}|\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})||, \\ & \max_{1\leqslant i<q\leqslant k} |\frac{1}{n}\sum_{t=1}^{n} |\phi(x_{tl_i}; a_i)\phi(x_{tl_q}; a_q)| - \mathbf{E}|\phi(X_{l_i}; a_i)\phi(X_{l_q}; a_i)||\}, \end{aligned} \tag{3.28}$$

where $a \in \mathcal{A}$.

We first prove that given $0 < \delta_1 < \frac{1-\xi}{2}$, there exists $C_1 > 0$ such that

$$\mathbf{P}(\tilde{d}(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) > n^{-\delta_1}) \leqslant \exp(-C_1 n^{1-2\delta_1}), \forall k \leqslant m. \tag{3.29}$$

Here we need the following lemma, which is proved by Ing & Lai (2011, Appendix A).

Lemma 1 *Let $\varepsilon, \varepsilon_1, \ldots, \varepsilon_n$ be i.i.d. random variables such that $\mathbf{E}(\varepsilon) = 0$, $\mathbf{E}(\varepsilon^2) = \sigma^2$ and $\mathbf{E}(\exp(s\varepsilon)) < +\infty$, $\forall |s| \leqslant s_0$ for some $s_0 > 0$. Then, for any constants a_{ni} $(1 \leqslant i \leqslant n)$ and $u_n > 0$ such that*

$$u_n \max_{1\leqslant i\leqslant n} \frac{|a_{ni}|}{A_n} \to 0 \text{ and } \frac{u_n^2}{A_n} \to +\infty \text{ as } n \to +\infty, \tag{3.30}$$

where $A_n = \sum_{i=1}^{n} a_{n_i}^2$, we have

$$\mathbf{P}(\sum_{i=1}^{n} a_{ni}\varepsilon_i > u_n) \leqslant \exp(\frac{-(1+o(1))u_n^2}{2\sigma^2 A_n}).$$

Proof of (3.29)**.** $\tilde{d}(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a)$ is defined as the maximum of 6 terms. We can use the same technique on each term. For example, for the first term, let

$$\varepsilon_t = \frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i})\phi(x_{tl_i}, a_i) - \mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i})\phi(X_{l_i}; a_i).$$

According to the assumptions in Section 3, all the conditions about ε_t in Lemma 1 are satisfied. Let $a_{ni} = 1$, $A_n = n$, $u_n = n^{1-\delta_1}$. Then it is easy to check that (3.30) holds. Therefore we can use Lemma 1 to show that

$$\mathbf{P}(|\frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i})\phi(x_{tl_i}; a_i) - \mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i})\phi(X_{l_i}; a_i)| > n^{-\delta_1}) < \exp(-Cn^{1-2\delta_1}),$$

where C is a constant that does not depend on n. Similar results hold for the other terms. Note that each of the terms is defined as the maximum of $k \leqslant m = o(\log(n))$ terms, we have

$$\mathbf{P}(\tilde{d}(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) = O(m\exp(-Cn^{1-2\delta_1})) \leqslant \exp(-Cn^{1-2\delta_1})$$

for some positive constant C.

(3.29) controls the probability of large deviation when the nonlinear parameter is limited at grid points. Now we consider the case where the nonlinear parameter is allowed to be around some fixed grid points. For sequences $\mathbf{s}_k = (s^{(1)}, \ldots, s^{(k)})$ and $\mathbf{a}_k = (a^{(1)}, \ldots, a^{(k)})$, we used the notation $\mathbf{s}_k \in I(\mathbf{a}_k)$ to define the event that

$$s_i \in I(a_i), i = 1, \ldots, k.$$

Define

$$
\begin{aligned}
d(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) \;=\; & \max\{ \max_{1\leqslant i\leqslant k, \mathbf{s}_k \in I(\mathbf{a}_k)} |\frac{1}{n}\sum_{t=1}^{n} \frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i})\phi(x_{tl_i}, s_i) - \mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i})\phi(X_{l_i}; s_i)|, \\
& \max_{\mathbf{s}_k \in I(\mathbf{a}_k), s\in I(a)} |\frac{1}{n}\sum_{t=1}^{n} \frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k})\phi(x_{tj}; s) - \mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k})\phi(X_j; s)|, \\
& \max_{1\leqslant i\leqslant k, s_i\in I(a_i)} |\frac{1}{n}\sum_{t=1}^{n} \phi^2(x_{tl_i}; s_i) - \mathbf{E}\phi^2(X_{l_i}; s_i)|, \\
& \max_{1\leqslant i<q\leqslant k, s_i\in I(a_i), s_q\in I(a_q)} |\frac{1}{n}\sum_{t=1}^{n} |\phi(x_{tl_i}; s_i)\phi(x_{tl_q}; s_q)| - \mathbf{E}|\phi(X_{l_i}; s_i)\phi(X_{l_q}; s_q)||\}.
\end{aligned} \tag{3.31}
$$

We will show that for $k \leqslant m$,

$$
\tilde{d}(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) \leqslant n^{-\delta_1} \Longrightarrow d(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) \leqslant 2n^{-\delta_1}. \tag{3.32}
$$

Assume $\tilde{d}(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) \leqslant n^{-\delta_1}$. Note that $\forall s_i \in I(a_i)$, $|s_i - a_i| \leqslant M_3 n^{-1}$. So we have for $i = 1, \ldots, k$,

$$
\begin{aligned}
& |\frac{1}{n}\sum_{t=1}^{n} \phi^2(x_{tl_i}; s_i) - \mathbf{E}\phi^2(X_{l_i}; s_i)| \\
= \;& |\frac{1}{n}\sum_{t=1}^{n} (\phi(x_{tl_i}; a_i) + (s_i - a_i)\phi'_{x_{tl_i}}(\eta_t))^2 - \mathbf{E}(\phi(X_{l_i}; a_i) + (s_i - a_i)\phi'_{X_{l_i}}(\eta))^2| \\
\leqslant \;& |\frac{1}{n}\sum_{t=1}^{n} \phi^2(x_{tl_i}; a_i) - \mathbf{E}\phi^2(X_{l_i}; a_i)| + 2(s_i - a_i)(|\frac{1}{n}\sum_{t=1}^{n} \phi(x_{tl_i}; a_i)\phi'_{x_{tl_i}}(\eta_t)| + |\mathbf{E}\phi(X_{l_i}; a_i)\phi'_{X_{l_i}}(\eta)|) \\
& + (s_i - a_i)^2(\frac{1}{n}\sum_{t=1}^{n} \phi'^2_{x_{tl_i}}(\eta_t) + \mathbf{E}\phi'^2_{X_{l_i}}(\eta)) \\
\leqslant \;& n^{-\delta_1} + 2n^{-1}M_3(M_4(M_2 + n^{-\delta_1}) + M_4M_2) + 2n^{-2}M_3^2M_4 \\
\leqslant \;& 2n^{-\delta_1}
\end{aligned}
$$

when n is large enough. Here the second inequality follows from the assumption

$\tilde{d}(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) \leqslant n^{-\delta_1}$ and (3.5), (3.8). Note that

$$f_{\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i} = \sum_{j=1}^{i} \rho_j \phi(X_j; s_j) = \sum_{j=1}^{i} \rho_j (\phi(X_j; a_j) + (s_j - a_j)\phi'_{X_j}(\eta_j)), \tag{3.33}$$

using (3.27), we have

$$|f_{\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i} - f_{\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i}| = |\sum_{j=1}^{i} \rho_j (s_j - a_j)\phi'_{x_j}(\eta_j)| \leqslant \frac{2i}{n} M M_3 M_4. \tag{3.34}$$

Similarly,

$$|f_{t,\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i} - f_{t,\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i}| \leqslant \frac{2i}{n} M M_3 M_4. \tag{3.35}$$

Let $\Delta L_i = \frac{\partial L}{\partial f}(y, f_{\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i}) - \frac{\partial L}{\partial f}(y, f_{\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})$. Then

$$|\Delta L_i| = |\frac{\partial^2 L}{\partial f^2}(y, \eta)(f_{\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i} - f_{\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})| \leqslant \frac{2i}{n} c_2 M M_3 M_4. \tag{3.36}$$

Let $\Delta L_{ti} = \frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i}) - \frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})$. Then we also have

$$|\Delta L_{ti}| \leqslant \frac{2i}{n} c_2 M M_3 M_4. \tag{3.37}$$

Let $\Delta\phi_i = \phi(x_{l_i}; s_i) - \phi(x_{l_i}; a_i)$ and $\Delta\phi_{ti} = \phi(x_{tl_i}; s_i) - \phi(x_{tl_i}; a_i)$. Then

$$|\Delta\phi_i| \leqslant \frac{2}{n} M_3 M_4, \quad |\Delta\phi_{ti}| \leqslant \frac{2}{n} M_3 M_4. \tag{3.38}$$

Therefore

$$
\begin{aligned}
& |\frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i})\phi(x_{tl_i}; s_i) - \mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{s}_i,\boldsymbol{\rho}_i})\phi(X_{l_i}; s_i)| \\
= \; & |\frac{1}{n}\sum_{t=1}^{n}(\frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i}) + \Delta L_{ti})(\phi(x_{tl_i}; a_i) + \Delta\phi_{ti}) \\
& -\mathbf{E}(\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i}) + \Delta L_i)(\phi(X_{l_i}; a_i) + \Delta\phi_i)| \\
\leqslant \; & |\frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})\phi(x_{tl_i}; a_i) - \mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})\phi(X_{l_i}; a_i)| \\
& +|\frac{1}{n}\sum_{t=1}^{n}\Delta L_{ti}\phi(x_{tl_i}; a_i)| + |\mathbf{E}\Delta L_i\phi(X_{l_i}; a_i))| \\
& +|\frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, f_{t,\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})\Delta\phi_{ti}| + |\mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_i,\mathbf{a}_i,\boldsymbol{\rho}_i})\Delta\phi_i| \\
& +|\frac{1}{n}\sum_{t=1}^{n}\Delta L_{ti}\Delta\phi_{ti}| + |\mathbf{E}\Delta L_i\Delta\phi_i| \\
\leqslant \; & n^{-\delta_1} + \frac{m}{n}c_2 M M_3 M_4(M_2 + n^{-\delta_1}) + \frac{1}{n}M_3 M_4 M_5 + \frac{c_2 M M_3^2 M_4}{n^2} \\
\leqslant \; & 2n^{-\delta_1}
\end{aligned}
\tag{3.39}
$$

when n is large enough. Here the second inequality follows directly from $\tilde{d}(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) \leqslant n^{-\delta_1}$ and assumptions (3.3), (3.5), (3.7) and (3.8). M_5 is an upper bound for $|\mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k})\phi(X_j; s)$ with arbitrary $\mathbf{l}_k \in \{1, \ldots, p\}^k$, $\mathbf{s}_k \in \Gamma^k$, $\boldsymbol{\rho}_k \in \mathbb{R}^k$, $j \in \{1, \ldots, p\}$ and $s \in \Gamma$. To see that upper bound exists, let

$$
\underline{f_{k,s}} = f_{\mathbf{l}_{k-1},\mathbf{s}_{k-1},\boldsymbol{\rho}_{k-1}} + \tilde{\pi}_k\phi(x_j; s),
$$

where

$$
\tilde{\pi}_k = \arg\min_{\tilde{\pi}} \mathbf{E}L(Y, f_{\mathbf{l}_{k-1},\mathbf{s}_{k-1},\boldsymbol{\rho}_{k-1}} + \tilde{\pi}\phi(X_j; s)). \tag{3.40}
$$

Taking derivative in $\tilde{\pi}$ and evaluating at $\tilde{\pi} = \tilde{\pi}_k$, we have

$$
\mathbf{E}\frac{\partial L}{\partial f}(Y, \underline{f_{k,s}})\phi(X_j; s) = 0.
$$

Hence

$$\begin{aligned}|\mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k})\phi(X_j;s)| &= |\mathbf{E}(\frac{\partial L}{\partial f}(Y,\tilde{f}_{\mathbf{w}_k}) - \frac{\partial L}{\partial f}(Y,\underline{f_{k,s}}))\phi(X_j;s)| \\ &= |\mathbf{E}\frac{\partial^2 L}{\partial f^2}(Y,\eta)(\rho_k\phi(X_{l_k};s_k) - \tilde{\pi}_k\phi(X_j;s))\phi(X_j;s)| \\ &\leqslant c_2|\mathbf{E}(\rho_k\phi(X_{l_k};s_k) - \tilde{\pi}_k\phi(X_j;s))\phi(X_j;s)|.\end{aligned}$$

Condition (3.27) claims that ρ_k is bounded. In the proof of Theorem 2, we have shown that the steps size $\tilde{\pi}_k$ will always be bounded as well. Therefore, we can use assumption (3.4) to prove the boundedness of $|\mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k})\phi(X_j;s)|$. So we claim that there exists $M_5 > 0$, such that

$$|\mathbf{E}\frac{\partial L}{\partial f}(Y, f_{\mathbf{l}_k,\mathbf{s}_k,\boldsymbol{\rho}_k})\phi(X_j;s)| \leqslant M_5, \forall \mathbf{l}_k \in \{1,\ldots,p\}^k, \mathbf{s}_k \in \Gamma^k, \boldsymbol{\rho}_k \in \mathbb{R}^k, j \in \{1,\ldots,p\}, s \in \Gamma. \tag{3.41}$$

We have shown that the first and third terms of $d(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a)$ are bounded by $n^{-\delta_1}$. The same argument can to applied to show that this result holds for the remaining terms as well. Therefore, (3.32) is true. So we have

$$\mathbf{P}(d(\mathbf{l}_k, \mathbf{a}_k, \boldsymbol{\rho}_k, j, a) > 2n^{-\delta_1}) \leqslant \exp(-C_1 n^{1-2\delta_1}), \forall k \leqslant m. \tag{3.42}$$

We have bounded the probability of large deviation for each variable and each subinterval of Γ. This result shows that the set of basis functions is redundant, in the sense that the difference between two basis functions whose nonlinear parameters are in the same subinterval can be controlled. Based on this result, we can start proving (3.22). First we compare $\tilde{\rho}_k$ and $\hat{\rho}_k$, which are defined by (3.19) and (3.21). Recall that

$$\mathbf{w}_k = ((l_1, s_1), \ldots, (l_k, s_k)).$$

Let $\mathbf{a}_k = (a_1, \ldots, a_k)$, where a_j is the grid point such that $s_j \in I(a_j)$, $j = 1, \ldots, k$. For $j \in 1, \ldots, p$, $a \in \mathcal{A}$ and $r \in I(a)$, we write $d(\mathbf{l}_m, \mathbf{s}_m, \tilde{\boldsymbol{\rho}}_m(\mathbf{w}_m), j, a)$ as d for simplicity.

According to (3.19), we have, by taking derivative of ρ, that

$$\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_{l_k}; s_k) = 0.$$

Similarly,

$$\frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k})\phi(x_{tl_k}; s_k) = 0.$$

Define $\tilde{f}_{t,\mathbf{w}_k}$ as the value of function $\tilde{f}_{\mathbf{w}_k}(\cdot)$ at $\mathbf{x}_t$. Then we have

$$\begin{aligned}
0 &= \frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k})\phi(x_{tl_k}; s_k) - \mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_{l_k}; s_k) \\
&= \frac{1}{n}\sum_{t=1}^{n}(\frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k}) - \frac{\partial L}{\partial f}(y_t, \tilde{f}_{t,\mathbf{w}_k}))\phi(x_{tl_k}; s_k) + \frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, \tilde{f}_{t,\mathbf{w}_k})\phi(x_{tl_k}; s_k) \\
&\quad -\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_{l_k}; s_k) \\
&= \frac{1}{n}\sum_{t=1}^{n}\frac{\partial^2 L}{\partial f^2}(y_t, \xi_t)(\hat{f}_{t,\mathbf{w}_k} - \tilde{f}_{t,\mathbf{w}_k})\phi(x_{tl_k}; s_k) + \frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, \tilde{f}_{t,\mathbf{w}_k})\phi(x_{tl_k}; s_k) \\
&\quad -\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_{l_k}; s_k) \\
&= \frac{1}{n}\sum_{t=1}^{n}\frac{\partial^2 L}{\partial f^2}(y_t, \xi_t)(\hat{\rho}_{\mathbf{w}_k} - \tilde{\rho}_{\mathbf{w}_k})\phi^2(x_{tl_k}; s_k) \\
&\quad +\frac{1}{n}\sum_{t=1}^{n}\frac{\partial^2 L}{\partial f^2}(y_t, \xi_t)\sum_{i=0}^{k-1}(\hat{\rho}_{\mathbf{w}_i} - \tilde{\rho}_{\mathbf{w}_i})\phi(x_{tl_i}; s_i)\phi(x_{tl_k}; s_k) \\
&\quad +\frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, \tilde{f}_{t,\mathbf{w}_k})\phi(x_{tl_k}; s_k) - \mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_{l_k}; s_k).
\end{aligned}$$

Rearranging terms, we get

$$\begin{aligned}
& \frac{1}{n}\sum_{t=1}^{n}\frac{\partial^2 L}{\partial f^2}(y_t,\eta_t)(\hat{\rho}_{\mathbf{w}_k}-\tilde{\rho}_{\mathbf{w}_k})\phi^2(x_{tl_k};s_k) \\
= & -\frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t,\hat{f}_{t,\mathbf{w}_k})\phi(x_{tl_k};s_k)+\mathbf{E}\frac{\partial L}{\partial f}(Y,\tilde{f}_{\mathbf{w}_k})\phi(X_{l_k};s_k) \\
& -\frac{1}{n}\sum_{t=1}^{n}\frac{\partial^2 L}{\partial f^2}(y_t,\eta_t)\sum_{i=i}^{k-1}(\hat{\rho}_{\mathbf{w}_i}-\tilde{\rho}_{\mathbf{w}_i})\phi(x_{tl_i};s_i)\phi(x_{tl_k};s_k).
\end{aligned}$$

Therefore

$$\begin{aligned}
c_1(m_1-d)|\hat{\rho}_{\mathbf{w}_k}-\tilde{\rho}_{\mathbf{w}_k}| &\leqslant d+\sum_{i=1}^{k-1}c_2|\hat{\rho}_{\mathbf{w}_i}-\tilde{\rho}_{\mathbf{w}_i}|\frac{1}{n}\sum_{t=1}^{n}|\phi(x_{tl_i};s_i)\phi(x_{tl_k};s_k)| \\
&\leqslant d+c_2(M_1+d)\sum_{i=1}^{k-1}|\hat{\rho}_{\mathbf{w}_i}-\tilde{\rho}_{\mathbf{w}_i}|.
\end{aligned}$$

On $\{d\leqslant 2n^{-\delta_1}\}$, $d<\frac{m_1}{2}$ when n is large enough. Therefore

$$\frac{c_1m_1}{2}|\hat{\rho}_{\mathbf{w}_k}-\tilde{\rho}_{\mathbf{w}_k}|\leqslant d+c_2(M_1+\frac{m_1}{2})\sum_{r=0}^{k-1}|\hat{\rho}_{\mathbf{w}_r}-\tilde{\rho}_{\mathbf{w}_r}|.$$

Let S_k be an upper bound for $|\hat{\rho}_{\mathbf{w}_k}-\tilde{\rho}_{\mathbf{w}_k}|$, such that

$$S_0=\frac{d}{c_1}$$

and

$$\frac{c_1m_1}{2}S_k\leqslant d+c_2(M_1+\frac{m_1}{2})\sum_{r=0}^{k-1}S_r.$$

It could be shown by induction that

$$S_k=\frac{2c_1+2c_2M_1+c_2m_1}{c_1^2m_1}(1+\frac{2c_2(M_1+\frac{m_1}{2})}{c_1m_1})^{k-1}d.$$

$$\therefore |\hat{\rho}_{\mathbf{w}_k} - \tilde{\rho}_{\mathbf{w}_k}| \leqslant \frac{2c_1 + 2c_2M_1 + c_2m_1}{c_1^2 m_1}(1 + \frac{2c_2(M_1 + \frac{m_1}{2})}{c_1 m_1})^{k-1}d. \tag{3.43}$$

Hence we have

$$\begin{aligned}
& |\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_j; r) - \frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k})\phi(x_{tj}; r)| \\
\leqslant\ & |\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_j; r) - \frac{1}{n}\sum_{t=1}^{n}\frac{\partial L}{\partial f}(y_t, \tilde{f}_{t,\mathbf{w}_k})\phi(x_{tj}; r)| \\
& + |\frac{1}{n}\sum_{t=1}^{n}(\frac{\partial L}{\partial f}(y_t, \tilde{f}_{t,\mathbf{w}_k}) - \frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k}))\phi(x_{tj}; r)| \\
\leqslant\ & d + |\frac{1}{n}\sum_{t=1}^{n}\frac{\partial^2 L}{\partial f^2}(y_t, \eta_t)(\tilde{f}_{t,\mathbf{w}_k} - \hat{f}_{t,\mathbf{w}_k})\phi(x_{tj}; r)| \\
\leqslant\ & d + |\frac{1}{n}\sum_{t=1}^{n}\frac{\partial^2 L}{\partial f^2}(y_t, \eta_t)(\sum_{i=1}^{k}(\hat{\rho}_{\mathbf{w}_i} - \tilde{\rho}_{\mathbf{w}_i})\phi(x_{tl_i}; s_i))\phi(x_{tj}; r)| \\
\leqslant\ & d + c_2\sum_{i=1}^{k}|\hat{\rho}_{\mathbf{w}_i} - \tilde{\rho}_{\mathbf{w}_i}|\frac{1}{n}\sum_{t=1}^{n}|\phi(x_{tl_i;s_i})\phi(x_{tj}; r)| \\
\leqslant\ & d + c_2(d + M_1)\sum_{i=1}^{k}|\hat{\rho}_{\mathbf{w}_i} - \tilde{\rho}_{\mathbf{w}_i}| \\
\leqslant\ & d + c_2(\frac{m_1}{2} + M_1)\sum_{i=1}^{k}|\hat{\rho}_{\mathbf{w}_i} - \tilde{\rho}_{\mathbf{w}_i}| \\
\leqslant\ & d + c_2(\frac{m_1}{2} + M_1)\sum_{i=1}^{k}\frac{2c_1 + 2c_2M_1 + c_2m_1}{c_1^2 m_1}(1 + \frac{2c_2(M_1 + \frac{m_1}{2})}{c_1 m_1})^{i-1}d \\
=\ & M_6(1 + \frac{2c_2(M_1 + \frac{m_1}{2})}{c_1 m_1})^k d,
\end{aligned}$$

where M_6 is a positive integer that does not depend on k.

It is not hard to show that for $a, b \in \mathbb{R}, c > 0$

$$(a - \frac{b}{c})^2 \leqslant \frac{2(a-b)^2}{c^2} + 2(\frac{1}{c} - 1)^2 a^2.$$

Therefore

$$\begin{aligned}
(\hat{\mu}_{\mathbf{w}_k,j,r} - \tilde{\mu}_{\mathbf{w}_k,j,r})^2 &= (\frac{\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_j;r)}{\sqrt{\mathbf{E}\phi^2(X_j;r)}} - \frac{\frac{1}{n}\sum_{t=1}^n \frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k})\phi(x_{tj};r)}{\sqrt{\frac{1}{n}\sum_{t=1}^n \phi^2(x_{tj};r)}})^2 \\
&= \frac{1}{\mathbf{E}\phi^2(X_j;r)}(\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_j;r) - \frac{\frac{1}{n}\sum_{t=1}^n \frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k})\phi(x_{tj};r)}{\sqrt{\frac{1}{n}\sum_{t=1}^n \phi^2(x_{tj};r)/\mathbf{E}\phi^2(X_j;r)}})^2 \\
&\leqslant \frac{2}{\frac{1}{n}\sum_{t=1}^n \phi^2(x_{tj};r)}(\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_j;r) - \frac{1}{n}\sum_{t=1}^n \frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k})\phi(x_{tj};r))^2 \\
&\quad + \frac{2}{\mathbf{E}\phi^2(X_j;r)}(\frac{1}{\sqrt{\frac{1}{n}\sum_{t=1}^n \phi^2(x_{tj};r)/\mathbf{E}\phi^2(x_j;r)}} - 1)^2(\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_{tj};r))^2 \\
&\leqslant \frac{4}{m_1}(\mathbf{E}\frac{\partial L}{\partial f}(Y, \tilde{f}_{\mathbf{w}_k})\phi(X_j;r) - \frac{1}{n}\sum_{t=1}^n \frac{\partial L}{\partial f}(y_t, \hat{f}_{t,\mathbf{w}_k})\phi(x_{tj};r))^2 \\
&\quad + M_5(\frac{\mathbf{E}\phi^2(X_j;r)}{\frac{1}{n}\sum_{t=1}^n \phi^2(x_{tj};r)} - 1)^2 \\
&= \frac{4}{m_1}M_6^2(1 + \frac{2c_2(M_1 + \frac{m_1}{2})}{c_1 m_1})^{2k}d^2 + \frac{8M_5}{m_1}d^2,
\end{aligned}$$

where M_5 is defined in (3.41). Therefore

$$\begin{aligned}
& \mathbf{P}((\hat{\mu}_{\mathbf{w}_k,j,r} - \tilde{\mu}_{\mathbf{w}_k,j,r})^2 > n^{-2\delta_1}) \\
\leqslant\ & \mathbf{P}(d^2 > \frac{n^{-2\delta_1}}{\frac{4}{m_1}M_3^2(1 + \frac{2c_2(M_1+\frac{m_1}{2})}{c_1 m_1})^{2k} + \frac{8M_6}{m_1}}) \\
\leqslant\ & \mathbf{P}(d > 2n^{-\delta_2}) \\
\leqslant\ & \exp(-C_2 n^{1-2\delta_2}).
\end{aligned}$$

for some $C_2 > 0$ and $\delta_2 \in (\delta_1, \frac{1-\xi}{2})$. The above inequality holds for fixed $\mathbf{w}_k$, j and r. We then have

$$\begin{aligned}
\mathbf{P}(\tilde{A}_n^c) &= \mathbf{P}(\max_{\mathbf{l}_k,\mathbf{a}_k,j,a} d(\mathbf{l}_k, \mathbf{s}_k, \tilde{\boldsymbol{\rho}}_k(\mathbf{w}_k), j, a) > n^{-\delta_2}) \\
&\leqslant (pn)^m \exp(-C_2 n^{1-2\delta_2}) \\
&= \exp(C_3 n^{\xi} - C_2 n^{1-2\delta_2} + O(\log^2(n))).
\end{aligned}$$

Hence (3.22) is proved.

Proof of (3.26)

We have

$$\mathbf{E}(\hat{f}_{\mathbf{v}_m}(\mathbf{X}) - \tilde{f}_{\mathbf{v}_m}(\mathbf{X}))^2 = \mathbf{E}(\sum_{k=1}^{m}(\hat{\rho}_k - \tilde{\rho}_k)\phi(X_{\hat{j}_k}; \hat{b}_k))^2.$$

Recall that $\hat{\mathbf{v}}_m = ((\hat{j}_1, \hat{b}_1), \ldots, (\hat{j}_m, \hat{b}_m))$ is the sequence of parameters selected by the sample version of the gradient boosting machine. Let

$$\hat{\mathbf{j}}_m = (\hat{j}_1, \ldots, \hat{j}_m),\ \hat{\mathbf{a}}_m = (\hat{a}_1, \ldots, \hat{a}_m),\ \hat{\boldsymbol{\rho}}_m = (\hat{\rho}_1, \ldots, \hat{\rho}_m),$$

where $\hat{a}_k \in \mathcal{A}$ and $\hat{b}_k \in I(\hat{a}_k)$ for $k = 1, \ldots, m$. Using the same argument as the proof of (3.43), we can show that

$$|\hat{\rho}_k - \tilde{\rho}_k| = O((1 + \frac{2c_2(M_1 + \frac{m_1}{2})}{c_1 m_1})^{k-1} d),$$

where d is short for $d(\hat{\mathbf{j}}_m, \hat{\mathbf{a}}_m, \hat{\boldsymbol{\rho}}_m, j, a)$ for arbitrary j and a. The definition of $d(\hat{\mathbf{j}}_m, \hat{\mathbf{a}}_m, \hat{\boldsymbol{\rho}}_m, j, a)$ can be found in (3.31).

Therefore

$$\begin{aligned}
& \mathbf{E}(\hat{f}_{\mathbf{v}_m}(\mathbf{X}) - \tilde{f}_{\mathbf{v}_m}(\mathbf{X}))^2 \\
= \;& \mathbf{E}(\sum_{k=1}^{m}(\hat{\rho}_k - \tilde{\rho}_k)\phi(X_{\hat{j}_k}; \hat{b}_k))^2 \\
= \;& \mathbf{E}(\sum_{k=1}^{m}(\hat{\rho}_k - \tilde{\rho}_k)\phi(X_{\hat{j}_k}; \hat{b}_k))^2\mathbf{1}_{\{d\leqslant 2n^{-\delta_1}\}} + \mathbf{E}(\sum_{k=1}^{m}(\hat{\rho}_k - \tilde{\rho}_k)\phi(x_{\hat{j}_k}; \hat{b}_k))^2\mathbf{1}_{\{d>2n^{-\delta_1}\}} \\
\leqslant \;& CM_1^2\sum_{k=1}^{m}\mathbf{E}|\hat{\rho}_k - \tilde{\rho}_k|^2\mathbf{1}_{\{d\leqslant 2n^{-\delta_1}\}} + CM_1^2\sum_{k=1}^{m}|\hat{\rho}_k - \tilde{\rho}_k|^2\mathbf{E}\mathbf{1}_{\{d>2n^{-\delta_1}\}} \\
\leqslant \;& \sum_{k=1}^{m} O(\mathbf{E}(1 + \frac{2c_2(M_1 + \frac{m_1}{2})}{c_1 m_1})^{2k-2}d^2\mathbf{1}_{\{d\leqslant 2n^{-\delta_1}\}}) \\
& + CM_1^2\sum_{k=1}^{m}(|\hat{\rho}_k|^2 + |\tilde{\rho}_k|^2)\mathbf{E}\mathbf{1}_{\{d>2n^{-\delta_1}\}} \\
= \;& O(\mathbf{E}(1 + \frac{2c_2(M_1 + \frac{m_1}{2})}{c_1 m_1})^{2m}d^2\mathbf{1}_{\{d\leqslant 2n^{-\delta_1}\}}) + CM_1^2\sum_{k=1}^{m}(|\hat{\rho}_k|^2 + |\tilde{\rho}_k|^2)\mathbf{E}\mathbf{1}_{\{d>2n^{-\delta_1}\}}.
\end{aligned}$$

The first term is $O(n^{-2\delta_1})$ since $m = o(\log(n))$. For the second term, note that we have shown in the proof of Theorem 2 that the steps sizes ρ_k calculated by the gradient boosting iterations are always summable, i.e.

$$\sum_{k=1}^{+\infty}|\hat{\rho}_k| < +\infty \;\text{ and }\; \sum_{k=1}^{+\infty}|\tilde{\rho}_k| < +\infty,$$

we then have

$$\sum_{k=1}^{m}(|\hat{\rho}_k|^2 + |\tilde{\rho}_k|^2) < +\infty \;\text{ as } m \to +\infty.$$

In (3.42), we showed that

$$\mathbf{E}\mathbf{1}_{\{d>2n^{-\delta_1}\}} = \mathbf{P}(d > 2n^{-\delta_1}) \leqslant \exp(-C_1 n^{1-2\delta_1}).$$

Hence (3.26) is proved.

3.5 Further Discussion on the Asymptotic Theory of the Gradient Boosting Machine

3.5.1 Multi-dimensional Parameters

In previous sections we only discussed the case where the nonlinear parameter r is in a one-dimensional interval $[\gamma_1, \gamma_2] \in \mathbb{R}$. Actually the theory can be easily extended to the case where $\mathbf{r}$ is a vector of dimension $d > 1$. Here we should assume that each dimension of $\mathbf{r}$ is bounded. So the assumption (3.7) becomes

$$r_j \in [\gamma_1^{(j)}, \gamma_2^{(j)}] \subset \mathbb{R},\ \gamma_2^{(j)} - \gamma_1^{(j)} \leqslant M_3 < \infty, j = 1, \ldots, d. \tag{3.44}$$

We also assume that the gradient of the basis function in $\mathbf{r}$ is bounded. We modify the assumption (3.8) to

$$\|\nabla \phi_{\mathbf{r}}(x_j; r)\|_\infty \leqslant M_4. \tag{3.45}$$

We divide the interval $[\gamma_1^{(j)}, \gamma_2^{(j)}]$ in to n subintervals for each j. So the set

$$[\gamma_1^{(1)}, \gamma_2^{(1)}] \times \cdots \times [\gamma_1^{(d)}, \gamma_2^{(d)}]$$

is divided into n^d subsets. For each subset, we can choose a grid point $\mathbf{a}$ (which will be a d-dimensional vector). Then for any other point $\mathbf{s}$ in the same subset. The difference of $\phi(x_j; \mathbf{s})$ and $\phi(x_j; \mathbf{a})$ can be bounded by using multi-dimensional Taylor expansion. Similar to (3.34), we have

$$|f_{\mathbf{l}_i, \mathbf{s}_i, \boldsymbol{\rho}_i} - f_{\mathbf{l}_i, \mathbf{a}_i, \boldsymbol{\rho}_i}| = |\sum_{j=1}^{i} \rho_j (\mathbf{s}_j - \mathbf{a}_j)^T \nabla(x_j; \boldsymbol{\eta}_j)| \leqslant \frac{2i}{n} M M_3 M_4. \tag{3.46}$$

The same result as (3.35) could also be established. So we can bound the difference of any linear combination of basis functions in the same subset. Since there are finite many subsets, we can follow exactly the same procedure as the one-dimensional case to establish the asymptotic theory.

3.5.2 Gradient Boosting Machine for the Regression Tree

As we have mentioned before, the regression tree with $\phi(x_{j_k}; r_k) = \mathbf{1}_{\{x_{j_k} \leqslant r_k\}}$ is one of the most commonly used nonlinear additive model. In the proof of the asymptotic theory, we require the base leaner to have bounded derivative, but the basis functions for the regression tree are not derivable in the nonlinear parameter r_k's. So we need to make a small modification to the proof. Again, we assume that $r \subset [\gamma_1, \gamma_2]$ and we divide this interval to n subintervals. We just need to bound the difference of basis functions in each subinterval, the remaining steps will be exactly the same as the derivable case.

Instead of dividing the interval $[\gamma_1, \gamma_2]$ equally into n subintervals, we divided it into $O(n)$ subintervals such that for any variable X_j, the probability that X_j is in any of these intervals is $O(\frac{1}{n})$. Let a be the grid point in an arbitrary subinterval and s be any point in the same interval, then for any variable X_j,

$$|\phi(X_j; s) - \phi(X_j; a)| = |\mathbf{1}_{\{X_j \leqslant s\}} - \mathbf{1}_{\{X_j \leqslant a\}}|$$

equals 1 if $X_j \in [\min(a, s), \max(a, s)]$ (whose probability is at most $O(\frac{1}{n})$), and 0 otherwise. Therefore

$$\mathbf{E}|\phi(X_j; s) - \phi(X_j; a)| = O(\frac{1}{n}).$$

Since the number of iterations m is assumed to be $o(\log(n))$, we can modify (3.34) as

$$\mathbf{E}|f_{\mathbf{l}_i, \mathbf{s}_i, \boldsymbol{\rho}_i} - f_{\mathbf{l}_i, \mathbf{a}_i, \boldsymbol{\rho}_i}| = o(\frac{\log(n)}{n}). \tag{3.47}$$

Instead of bounding the absolute value of $f_{\mathbf{l}_i, \mathbf{s}_i, \boldsymbol{\rho}_i} - f_{\mathbf{l}_i, \mathbf{a}_i, \boldsymbol{\rho}_i}$, we are bounding its absolute expectation. Similarly, we can modify the boundedness for $|\Delta L_i|$ and $|\Delta \phi_i|$ (inequalities (3.36), (3.37) and (3.38)) to the boundedness of expectation (or sample average), and can follow the same procedure to prove the boundedness (3.39). Then we can proceed to prove the convergence and consistency, just as the derivable case.

In the next chapter, we will illustrate the performance of the gradient boosting machine on fitting the tree model through a simulation study.

3.6 Stopping Criteria for the Gradient Boosting Machine

Theorem 1 states that as the size of the training samples n and the number of iterations m go to infinity, the model $\hat{f}^m(\cdot)$ fitted by the gradient boosting machine converges to the real model $f(\cdot)$ in the L_2 norm. In practice, for a fixed sample size n, we cannot iterate too much, as that would make the model too complex and affect its prediction performance. Instead, we need a rule to stop the iteration earlier to ensure the accuracy or interpretation of the model. There are various stopping criteria for the gradient boosting machine. Our choice should coincide with the purpose for building the model.

Recall that the gradient booting machine adds variables (basis functions) to the model according to their correlation to the current descent direction, the major contributors (most relevant variables/basis functions) are added during the first few iterations. So if our goal is model selection, then we could stop the iteration when the number of variables (basis functions) in the model reaches a desired level. Through this stopping criterion we will sacrifice some accuracy, but we will be able capture the main structure of the relationship between the input $\mathbf{X}$ and output Y. In addition, the resulting model will be very concise.

On the other hand, if our purpose is to obtain an accurate model, then the stopping criterion should be based on the expected prediction error.

$$\mathbf{E}L(Y, \hat{f}^k(\mathbf{X})) \tag{3.48}$$

Of course, there is no way for us to calculate the exact value of (3.48) for each step k, since the joint distribution of $\mathbf{X}$ and Y is unknown. Therefore, we need some method to estimate it and should stop the iteration when the estimated prediction error is small enough.

One of the candidates for the estimation is the training error

$$\frac{1}{n}\sum_{t=1}^{n} L(y_t, \hat{f}^k(\mathbf{x}_t)). \tag{3.49}$$

That is, the average of the loss function evaluated on all the training samples. Note that this training error is monotone decreasing as the iteration goes on, it should converge to zero or some positive number. Therefore, when the number of iterations is getting larger and larger, the progress we can make in terms of the training error is being less and less. So we can stop the iteration when the progress is lower than some threshold level. This criterion will include some irrelevant variables, but the accuracy of the model will in general be better than the previous criterion.

However, it is well known that the training error (3.49) is biased downwards as an estimator of the expected prediction error (3.48). A model that fits the training data "too well" will have poor prediction performance when being applied to other independent data sets, i.e. it will be *over fitting*. In contrast, ten-fold cross validation successfully fixes the bias of the training error, and achieves an excellent bias-variance tradeoff as an estimator of the prediction error. Therefore, it is widely used as a criterion for choosing the tuning parameters for various algorithms.

In the next chapter, we will go over more details of cross validation, and will use numerical examples to illustrate the performance of the gradient boosting machine. Then we compare the effect of this algorithm on variable selection and prediction for different stopping rules.

Chapter 4

Implementations and Simulation Studies of the Gradient Boosting Machine

In the previous chapter, we have established the asymptotic theory for the gradient boosting machine. Now we will implement this algorithm on simulated data to illustrate its performance. A practical problem for the gradient boosting machine is when to stop the iteration. Although Theorem 1 shows that as the sample size n and number of iterations m go to infinity, the model fitted by the gradient boosting machine converges to the real model, in practice we do not want to iterate too much, since that will result in problem of overfitting. So we need some criterion to stop the iteration as soon as the fitted model is accurate enough. The most commonly used method for determining the optimal number of iterations is cross validation.

In this chapter, we first review previous studies on cross validation, show how it estimates the prediction error consistently by compromising before bias and variance, and discuss its application as a stopping criterion for the gradient boosting machine. Then we use simulated data to illustrate the convergence and consistency of the gradient boosting machine, and compare the results for different stopping rules. Our first example is a linear model, where we show how to adjust the stopping rules of the

boosting algorithm to satisfy different purposes, e.g. excluding irrelevant variables or building an accurate model. We compare the result of the gradient boosting machine to the regularization method, and discuss the natures of these two algorithms. The second example is a nonlinear logistic regression model, where the regularization method fails to have a solution. We will illustrate the effectiveness of the gradient boosting machine on this problem.

4.1 Estimating the Prediction Error: Cross Validation

If our goal is to construct an statistical model with good prediction performance, then ideally, we wish to find a tuning parameter such that the corresponding model has the smallest prediction error. Let

$$T_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$$

be a set of n training samples. Let $\hat{f}_n^{(M)}(\cdot)$ a model fitted on T_n through some algorithm M. The optimal M has expected prediction error

$$\mathrm{Err}^{(M)} = \mathbf{E}L(Y, \hat{f}_n^{(M)}(\mathbf{X})) = \min_f \mathbf{E}L(Y, f(\mathbf{X})),$$

where the expectation is taken on the joint distribution $\mathcal{D}$ of the pair $(\mathbf{X}, Y)$.

In general, there is no way to calculate the exact value of the expected prediction error of $\hat{f}_n^{(M)}(\cdot)$, since the $\mathcal{D}$ unknown. Instead, we need to find alternative ways to estimate the expected prediction error. A natural candidate estimator is the training error

$$\mathrm{err} = \frac{1}{n}\sum_{t=1}^{n} L(y_t, \hat{f}_n^{(M)}(\mathbf{x}_t)), \tag{4.1}$$

which is the arithmetic average of values of the loss function on all the training samples. Unfortunately, the training error is not a good estimator of the real prediction error. The reason is simple: If we train and test the model using the same set of data,

the model "knows" the data before being tested and therefore have unfair advantage. In fact, the training error is usually an overly optimistic estimate of the real prediction error. Models that fit the training data "too well" may behave very poorly in predicting future data. This problem is referred as "overfitting".

For fixed training set T_n and loss function L, define the *optimism* as the difference between the expected prediction error Err and training error err. That is,

$$\text{op} = \text{op}(T_n) = \text{Err} - \text{err}.$$

In other words, $\text{op}(T_n)$ is the amount by how much the training error underestimates the real prediction error. Let ω be the expectation of $\text{op}(T_n)$:

$$\omega = \mathbf{E}\text{op}(T_n),$$

where the expectation is taken on the training set T_n. Again, in most cases, ω is unknown due to lack of knowledge of the distribution $\mathcal{D}$. So it has to be estimated from the training samples. Let $\hat{\omega}$ be an estimator of ω. The corresponding estimator of Err will be defined as

$$\widehat{\text{Err}} = \hat{\omega} + \text{err}.$$

4.1.1 Overview of Cross Validation

In order to estimate the prediction error Err consistently, we need some method to fix the bias of the training error err (which is defined as $\hat{\omega}$ in the above equation). One of the most commonly used techniques is Cross validation (CV). As is discussed above, constructing and evaluating a model based on the same set of data would result in an overoptimistic estimate of the prediction performance. Cross validation fixes this problem by separating the training and test sets so that the model won't "see" the test data before it is being tested. Suppose that we have n samples. The cross validation technique randomly divides the samples into two sets: a training set which contains n_t samples, and a validation set which contains $n_v = n - n_t$ samples. The model is constructed by using the training set, and the prediction error is estimated

by using the validation set. The final estimation of the prediction error is obtained by averaging the errors from all possible splits of the data set, i.e.

$$\widehat{\mathrm{Err}}^{(\mathrm{MCV})} = [n_v \begin{pmatrix} n \\ n_v \end{pmatrix}]^{-1} \sum_{S \text{ being a subset of size } n_v} \|y_S - \hat{f}^{-S}(X_S)\|_2^2, \qquad (4.2)$$

where X_S and y_S are input and response values of the samples in the validation set S. $\hat{f}^{-S}$ is the model fitted from the training set S^C.

Among all the possible choices for n_v, the setting of $n_v = 1$ leads to the simplest version of cross validation (leave-one-out CV), as it requires the least amount of computation. Leave-one-out CV was introduced by Allen (1974), Stone (1974) and Geisser (1975). In the case of n goes to infinity while the number of features p is fixed, the leave-one-out CV estimator is nearly unbiased for the real prediction error. However, Efron (1983) & (1986) points out that the leave-one-out CV is not a good estimator of Err since the *mean square error*

$$\mathrm{MSE}^{(\mathrm{CV})} = \mathbf{E}(\hat{\mathrm{Err}}^{(\mathrm{CV})} - \mathrm{Err})^2.$$

is usually high. The reason is not hard to explain in terms of the concept of optimism described above. For leave-one-out CV, the expected optimism is estimated as

$$\omega^{(CV)} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}^{-i}(X_i)) - \mathrm{err} = \hat{\mathrm{Err}}^{(CV)} - \mathrm{err},$$

where $\hat{f}^{-i}$ is the model fitted from all the training samples except the i-th one. From its formulation we see that the training sets used by leave-one-out CV to fit the model $\hat{f}^{-i}$'s are too close to the original dataset, which may not be a good representation of its real distribution. So $\hat{\mathrm{Err}}^{(CV)}$ would have high variance. Consequently, the variance of $\omega^{(CV)}$ would be high. As a result, the mean squared error $\mathrm{MSE}^{(\mathrm{CV})}$ is high as well, since equation (3.1) of Efron (1983) shows that $\mathrm{Var}(\omega^{(CV)})$ is an important component of $\mathrm{MSE}^{(\mathrm{CV})}$.

Therefore, in order to improve the performance of the $\hat{Err}^{(MCV)}$ in estimating Err,

we need to reduce its variance with the training set T_n treated as random. A natural idea is to increase the size of the validation set. This results in the delete-d CV, where $d = n_v$ and $d \gg 1$. Intuitively, when d gets very large (e.g. comparable to the sample size n), the size n_t of the training set would be very small, and the training set in one training-validation split would be quite different from another. By taking average of models trained by those data sets, the variance of the estimate could be controlled. On the other hand, if we require $n_t \to \infty$, as p is fixed, delete-d CV estimator will still be approximately unbiased of the real prediction error. Shao (1993) showed that in order for the cross validation estimator $\hat{Err}^{(MCV)}$ to be consistent with Err, n_v and n_t must satisfy $\frac{n_v}{n} \to 1$ and $n_t \to \infty$ as $n \to \infty$.

The problem of delete-d CV is that it is computationally infeasible when d is large. In practice, only a much smaller part of the $\begin{pmatrix} n \\ n_v \end{pmatrix}$ splits of the train and validation sets are made to reduce the computational cost. Breiman, Friedman, Olshen & Stone (1984) introduced the k-fold cross validation, which divides the data into k roughly equal-sized subsets, uses each subset in turn as the validation set and the remaining as the training set. Let S_j be the j-th subset. Let $\hat{f}^{-j}(x)$ be the fitted model based on the data after removing the j-th subset, then the k-fold cross validation error estimate is defined as

$$\widehat{\mathrm{Err}}^{(\mathrm{kCV})} = \frac{1}{k}\sum_{j=1}^{k}\frac{1}{|S_j|}\|y_{S_j} - \hat{f}^{-j}(X_{S_j})\|_2^2, \tag{4.3}$$

Another alternative to delete-d CV is the *Repeated Learning Test* (RLT), which was introduced by Breiman et al. (1984). It is also called the *Monte Carlo Cross Validation* (MCCV) in some other literatures. Instead of summing over all possible splits with $n_v = d$, MCCV just takes B random subsets $S_1^*, \ldots, S_B^*$ of size d and estimate the prediction error by

$$\widehat{\mathrm{Err}}^{(\mathrm{MCCV})} = \frac{1}{Bd}\sum_{i=1}^{B}\|y_{S_i^*} - \hat{f}^{-S_i^*}(X_{S_i^*})\|_2^2. \tag{4.4}$$

Zhang (1993) shows that $\widehat{\text{Err}}^{\text{kCV}}$ is asymptotically equivalent to $\widehat{\text{Err}}^{\text{MCV}}$ as $n \to +\infty$ under certain conditions.

According to Hastie et al. (2009), for k-fold cross validation, when the number of folds k is small, "bias could be a problem, depending on how the performance of the learning method varies with the size of the training set." Figure 4.1 plots a hypothetical learning curve for some learning method. In general, the prediction performance should increase with the size n of the training samples, as is reflected by this graph. Now suppose $n = 300$. If $k = 2$, then we are only using half of the training samples to construct the model, and use the other half as the validation set. In this case, the estimated prediction error will be close to the prediction error of the model fitted by using $300/2 = 150$ training samples. According to Figure 4.1, this estimator is seriously biased.

On the other hand, if k is too large, then the performance of the k-fold cross validation estimator will be close to leave-one-out cross validation, which has large variance. Therefore, k should be chosen to achieve a compromise between bias and variance. Although problems are quite different from each other, in general, the choice $k = 10$ is widely used in practice. The effectiveness of 10-fold cross validation has been proved by many examples.

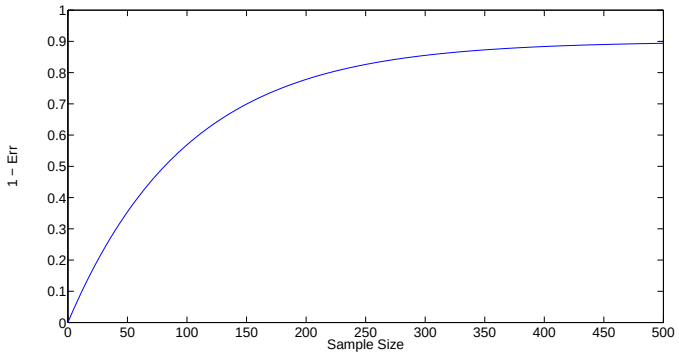

Figure 4.1: Hypothetical learning curve for some learning method

4.1.2 Cross Validation as a Stopping Criterion for the Gradient Boosting Machine

We can use 10-fold cross validation to choose a proper number of iterations for the gradient boosting machine. Here is the procedure: divide the training set T randomly into 10 subsets $T_1, \ldots, T_{10}$, which have roughly the same size. For $i = 1, \ldots, 10$, implement the gradient boosting machine on the set $T \backslash T_i$ with m iterations, where m is an pre-specified integer that is large enough. For $k = 1, \ldots, m$, let $\hat{f}^k_{-i}(\cdot)$ be the model fitted by the gradient boosting machine at the k-th iterations on the set $T \backslash T_i$. We use the average test error on $\hat{f}^k_{-i}(\cdot)$ to estimate the prediction error $\mathrm{Err}^{(k)}$ at the k-th iteration, i.e.

$$\mathrm{Err}^{(k)} = \frac{1}{10} \sum_{i=1}^{10} \frac{1}{|T_i|} \sum_{(\mathbf{x}_t, y_t) \in T_i} L(y_t, \hat{f}^k_{-i}(\mathbf{x}_t)).$$

The optimal number of iterations k^* is the one such that the above estimated prediction error is minimized. That is,

$$k^* = \arg \min_{k \in \{1, \ldots, m\}} \mathrm{Err}^{(k)}.$$

Once k^* is identified, we implement the gradient boosting machine again on all the training samples with k^* iterations, and output $\hat{f}^{k^*}(\cdot)$ as the final result.

In the next two sections, we use numeral examples to illustrate of the gradient boosting machine equipped with this stopping criterion.

4.2 Linear Regression

Consider a linear regression model

$$Y = \alpha + \boldsymbol{\beta}^T \mathbf{X} + \varepsilon. \tag{4.5}$$

Here $\mathbf{X} = (X_1, \ldots, X_p)$ is an random vector of length $p = 1000$, and

$$X_j = D_j + W, \quad j = 1, \ldots, p,$$

where $D_1, \ldots, D_p$ are i.i.d. normal random variables with mean 1 and variance 1, W is a normal random variable with mean 0 and variance 1, and is independent of D_j, $j = 1, \ldots, p$. So the covariates X_j's share the same value of W, and thus are correlated with each other. The coefficient vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is defined as the following:

$$(\beta_1, \ldots, \beta_9) = (3.2, 3.2, 3.2, 3.2, 4.4, 4.4, 3.5, 3.5, 3.5), \quad \beta_j = 0, \text{for } j \geqslant 10.$$

In addition, define $\alpha = 2$ as the intercept and $\varepsilon \sim \mathcal{N}(0, 1)$ as the random noise. So the model is high-dimensional but sparse.

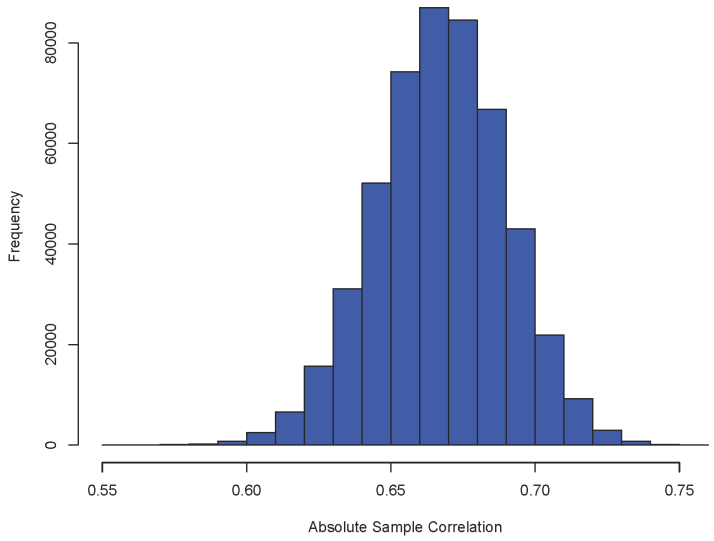

Figure 4.2: Histogram of absolute value of sample correlations among covariates

We generate $n = 300$ random training samples from the distribution described above, and calculate the absolute value of the sample correlation among the covariates.

Figure 4.2 plots the histogram of these sample correlations. It can be seen that these covariates are highly correlated with each other. This adds difficulty to variable selection. We will see how the gradient boosting machine performs under this high correlation setting.

4.2.1 Performance of Gradient Boosting Machine

We use quadratic loss $L(y, f(x)) = \frac{1}{2}(y - f(x))^2$ as the loss function, and define the basis function as $\phi(x_j; r) = x_j + r$. Here the parameter r is used to absorb the intercept α in model (4.5). It is easy to check that all the assumptions mentioned in Section 2.1 are satisfied. We set the number of iterations m as 50. For $k = 1, \ldots, m$, let $\hat{f}^k(\cdot)$ be the model obtained at the k-th iteration. Define the training error as

$$\text{err}_k = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{f}^k(\mathbf{x}_t))^2.$$

We plot the training error vs the number of iterations in Figure 4.3.

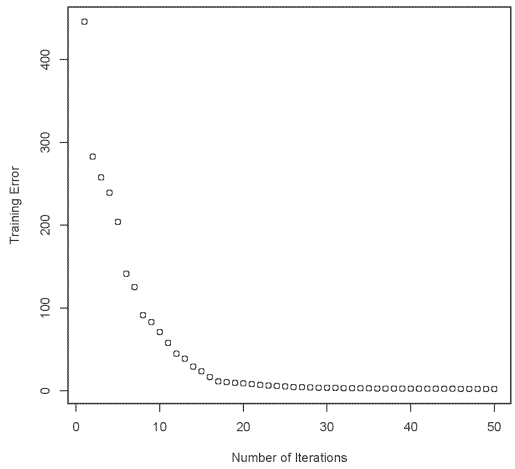

Figure 4.3: Training error vs number of iterations

It is seen that the training error decreases very rapidly at the beginning, and is close to 0 within 30 iterations. Then the curve becomes flat. This shows that most

Iteration Number	1	2	3	4	5	6	7	8	9	10
Covariate Index	6	5	1	5	6	4	6	9	6	7
Iteration Number	11	12	13	14	15	16	17	18	19	20
Covariate Index	6	3	5	8	6	2	4	912	2	6
Iteration Number	21	22	23	24	25	26	27	28	29	
Covariate Index	5	8	5	7	7	4	303	6	2	

Table 4.1: Covariates chosen by the first 29 iterations

contribution was made by the first few iterations. Based on this observation, we would not take too many iterations, as there will be very little progress in terms of the training error, and the resulting model could be overfitting. Instead, we choose the number of iterations k^* by 10-fold cross validation, which gives $k^* = 29$. To test the prediction performance of the model $\hat{f}^{29}(\cdot)$, we generate a test set of size $N = 5000$, and evaluate the test error

$$\text{Err}_{29} = \sum_{t=1}^{N} (y_t - \hat{f}^{29}(\mathbf{x}_t))^2.$$

In this example, we have $\text{Err}_{29} = 7.84$.

Table 3.1 lists the indices of covariates selected by each of the first 29 iterations. It's seen that for most of the iterations, the gradient boosting machine chose relevant variables. There are only two irrelevant variables (i.e. X_{912} and X_{303} included in our final result). On the other hand, if our focus is on variable selection, and we know beforehand that there are only 9 relevant variables, we can stop the iteration when the number of variables included in the model exceeds 9, i.e. we could stop at the 17-th iteration in this example. Then we will include all the relevant variables and no irrelevant variable. The corresponding test error is

$$\text{Err}_{17} = \sum_{t=1}^{N} (y_t - \hat{f}^{17}(\mathbf{x}_t))^2 = 12.52.$$

We excluded irrelevant variables with the price of a 60% increase in the prediction error.

4.2.2 Comparing with Lasso

We now compare the result of the gradient boosting machine to the regularization method. Here we use the `R` function `glmnet` and set the parameter `alpha`=1 to implement the Lasso penalty. The function `glmnet` automatically generates a list of tuning parameters λ. Starting from the largest one, which corresponding to the most sparse model, `glmnet` adds more variables to the model, and moves to smaller tuning parameters iteratively through a coordinate descent approach proposed by Friedman et al. (2008). When the algorithm terminates, it outputs the regression coefficients for each of the tuning parameters.

The optimal tuning parameter λ^* could be identified by 10-fold validation as well. For this example, we have $\lambda^* = 0.27$. Let $\hat{f}_{\lambda^*}(\cdot)$ be the model fitted by lasso with tuning parameter λ^*. The test error of $\hat{f}_{\lambda^*}(\cdot)$ is 6.74. This result is better than the gradient boosting machine, but $\hat{f}_{\lambda^*}(\cdot)$ contains 36 variables, 27 of which are irrelevant. So the interpretation of $\hat{f}_{\lambda^*}(\cdot)$ is much worse than $\hat{f}^{27}(\cdot)$, which was fitted by the gradient boosting machine.

This difference can be explained by the natures of the two algorithms. Lasso (as well as other regularization methods) fits the model "globally". The penalty $P(\boldsymbol{\beta})$ is added to all entries of the coefficient vector $\boldsymbol{\beta}$, so the numerical optimization technique that solves the penalized optimization problem (2.14) will move a group of coefficients simultaneously during each iteration. This may include irrelevant variables, especially when the correlation between relevant and irrelevant variables is high. In contrast, the gradient boosting machine fits the model "greedily". During each iteration, it only selects one variable, and fits the best result on it. Therefore, only a few variables would be included in the model. However, this greedy nature makes the algorithm move too far on some variables, thereby affecting the prediction performance of the resulting model.

The difference between these two algorithms can also be seen from the side of variable selection. For lasso, the smallest tuning parameter that allows 9 variables in the resulting model is 3.04. Although all the variables in the model $\hat{f}_{3.04}(\cdot)$ are relevant variables, the test error of this model is 24.35, which is twice as the test

error of $\hat{f}^{17}(\cdot)$. The big tuning parameter 3.04 can be viewed as an early stage of the lasso iterations, where relevant variables are just added, but not fully regressed. In contrast, the gradient booting machine "explains" most of its correlations to the output Y as soon as a new variable is added to the model. So if we only focus on these "early stage variables", the gradient boosting machine will have much better results.

Lasso, as well as other regularization methods, has been proved successful in fitting high-dimensional linear models. Our simulation result also shows that the model fitted by lasso has better prediction performance than the gradient boosting machine. However, lasso has a bad reputation for variable selection when there are high correlations among the variables. On the other hand, the greedy nature of the gradient boosting machine provides another insight for constructing the model. By focusing on a smaller subset of variables, the gradient boosting machine is able to fit a concise model with moderate prediction performance.

For regularization methods, the iterations for solving the penalized optimization problem (2.14) is getting more and more complex, as more and more variables are added to the model. In contrast, during each iteration of the gradient boosting machine, the progress is always made on a single variable (basis function). So the complexity will not increase as the iteration goes on. This allows the gradient boosting machine to be extended to more complicated (nonlinear) models.

4.3 Gradient Boosting Machine on the Nonlinear Logistic Regression Model

We now consider a nonlinear logistic regression model. Suppose Y is a Bernoulli random variable whose distribution depends on the regression function $f(\mathbf{X})$:

$$Y \sim \text{Bernoulli}(\frac{\exp(f(\mathbf{X}))}{1+\exp(f(\mathbf{X}))}). \tag{4.6}$$

The regression $f(\cdot)$ is a linear combination of indicator functions

$$f(\mathbf{X}) = \sum_{j=1}^{p} \beta_j \mathbf{1}_{\{X_j \geqslant r_j\}} + \varepsilon. \tag{4.7}$$

Again, we assume that the model is high-dimensional but sparse. Let

$$p = 600, \quad \beta_{1-6} = (1, 0.8, 0.9, -0.9, -0.9, -0.9)^T, \quad \beta_j = 0 \quad \text{for} \quad j > 6.$$

We also assume that each of the covariates $X_j \in [0, 1]$, and the nonlinear parameters

$$r_{1-6} = (0.61, 0.55, 0.37, 0.74, 0.26, 0.48).$$

The random noise ε is assumed to be $\mathcal{N}(0, 0.01)$.

This model could be applied to biology or clinical trials, where each of the genes is known to be above some threshold value (or not), and we want to predict some binary outcome (e.g. if a person has some disease).

Suppose we have $n = 400$ samples. In this low sample sized and high-dimensional case, the traditional algorithm for fitting the logistic regression model fails to converge. In addition, there is no way to apply regularization methods, because of the nonlinear parameter r_j in the basis function. Instead, we apply the gradient boosting machine to this problem.

As for the regular logistic regression model, we use the exponential loss (1.4) as the loss function. At each iteration, we use the `R` function `optimize` to maximize the squared sample correlation of $\mathbf{1}_{\{X_j \geqslant r\}}$ with the negative gradient as a univariate function of r for each covariate X_j, and we choose the covariate X_j and the corresponding parameter r_j such that the basis function $\mathbf{1}_{\{x_j \geqslant r\}}$ has the largest squared sample correlation with the negative gradient.

After selecting the optimal basis function, we calculate the step size ρ with this basis function. However, to make the algorithm more stable, instead of using the full step size ρ, we multiply it by a factor $\nu = 0.1$. We take this truncated step size to prevent us from going too far away in a possibly wrong direction.

Initially, we set the number of iterations m to be 150. For each $k \leqslant m$, we use 10-fold cross validation to estimate the prediction error of the model $\hat{f}^k(\cdot)$ obtained at the k-th interation. Instead of evaluating the exponential loss, we calculate the error rate

$$E_i^k = \frac{1}{|T_i|} \sum_{(\mathbf{x}_t, y_t) \in T_i} \mathbf{1}_{\{y_t \neq \mathbf{1}_{\{\hat{f}_{-i}^k(\mathbf{x}_t) \geqslant 0\}}\}}$$

for each fold. Here the T_i's are the random subsets of the training set, and $\hat{f}_{-i}^k(\cdot)$ is the model fitted at the k-th iteration without using T_i, i = 1,...,10.

We use the average

$$E^k = \frac{1}{10} \sum_{i=1}^{10} E_i^k$$

to estimate the error rate that corresponds to the iteration number k. We choose the number k^* with the smallest estimated error rate as the optimal number of iterations. Then we iterate our algorithm for k^* times on all the training samples, and output the corresponding model $\hat{f}^{k^*}(\cdot)$ as our final result.

To test the prediction performance of the resulting model, we generate a test set with size $N = 1000$. For $k = 1, \ldots, m$, we estimate the mean squared error (MSE) $\mathbf{E}(\hat{f}^k(\mathbf{X}) - f(\mathbf{X}))^2$ by using the sample mean on the test set. That is,

$$\mathbf{E}(\hat{f}^k(\mathbf{X}) - f(\mathbf{X}))^2 \approx \frac{1}{N} \sum_{t=1}^{N} (\hat{f}^k(\mathbf{X}_t) - f(\mathbf{X}_t))^2.$$

Figure 4.4 plots the MSE vs the number iterations. The red point is the optimal number of iterations selected by cross validation. We see that the MSE at that point is very close to the smallest one on the graph, and the curve after that point is flat. So it is a wise choice to stop iteration at that point to prevent the model from being too complex, which does not necessarily improve the accuracy of prediction.

Figure 4.5 plots the prediction error vs the number of iterations. Here the prediction error is estimated from the test set via

$$\frac{1}{N} \sum_{t=1}^{N} \mathbf{1}_{\{y_t \neq \hat{y}_t\}},$$

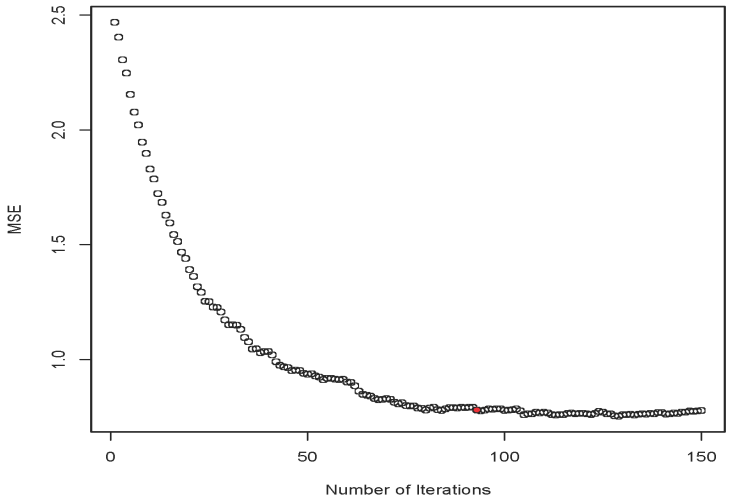

Figure 4.4: Convergence of the regression function

where $\hat{y}_t$ is the predicted value of the model $\hat{f}^k(\cdot)$, i.e.

$$\hat{y}_t = \mathbf{1}_{\{\hat{f}^k(\mathbf{x}_t)>0\}}, \quad t = 1, \ldots, N.$$

It's seen that the error rate decreases very rapidly during the first few iterations, then the speed of decrease becomes approximately linear. When the error rate reaches the lowest level, it goes back up a little bit. This shows that too many iterations will result in overfitting. As in Figure 4.4, the red point is the stopping position selected by 10-fold cross validation. Again, it is very close to the minimal value on the graph. For this problem, the Bayesian error is 0.21, and our result is just two percentage points above it. Considering the high dimensionality and nonlinearity of this problem, our result is reasonable. This example shows that the gradient boosting machine is effective in solving high-dimensional nonlinear modeling problems.

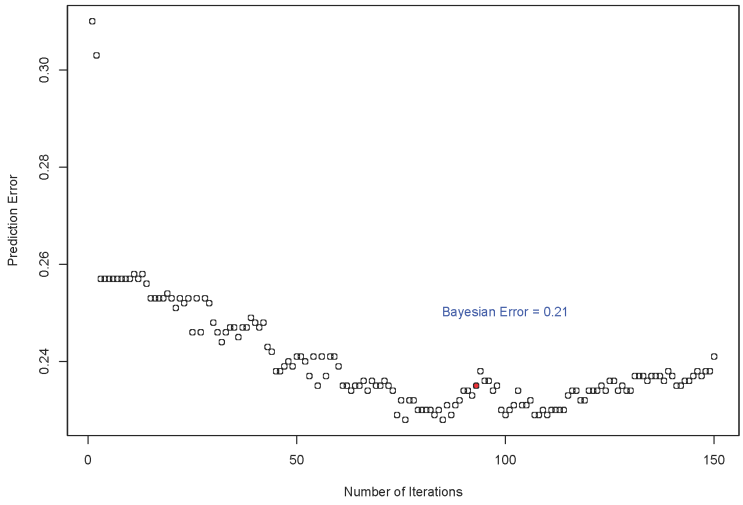

Figure 4.5: Convergence of the prediction error

Chapter 5

Conclusion

The additive model represents a broad class of statistical models. Examples such as linear models and regression trees have been widely used in practice. The gradient boosting machine is an efficient algorithm for fitting general additive models. Various examples have shown its effectiveness in many applications, and people have proved its convergence and consistency when being applied to linear models. However, the asymptotic properties of this algorithm on general nonlinear models, especially in the high-dimensional case where the number of covariates p far exceeds the sample size n, are still unclear. The major difficulty in exploiting this area is that the nonlinear parameters result in an infinite set of basis functions, making it hard to bound the probability of getting large difference between the expected value of the loss function and its sample mean estimation.

In this thesis, we addressed this difficulty by discretizing the set of basis functions and bounding the difference within each discretized segment. We showed that the set of basis functions is highly redundant. That is, the properties of two basis functions are very similar to each other when their nonlinear parameters are close. This result allows us to treat the infinite set of basis functions as finite. We considered an ideal case, namely the population version of the gradient boosting machine, whose asymptotic properties can be easily shown by using quadratic function approximation. Then we used the large deviation theory to control the difference between the outputs of the sample (practical) version and the population version of the algorithm. Based

on this result, we proved that as the number of iterations m and sample size n goes to infinity, with the number of covariates p growing no faster than exponentially with n, the model generated by the gradient boosting machine converges to the real model in the squared norm. This is the main contribution of this thesis.

In practice, it is very important to choose a proper stopping criterion for the gradient boosting machine. Iterating too much will result in the problem of overfitting, thereby damaging the prediction performance of the output model. In this thesis, we discussed the properties of the 10-fold cross validation method for estimating the prediction error, and illustrated how this method could be applied to select the number of iterations for the gradient boosting machine. Then we illustrated the performance of the gradient boosting machine equipped with this stopping criterion through simulation study.

We compared the performance of gradient boosting machine and the regularization method on a linear model. Based on the difference of the results, we discussed the natures of these two algorithms. The regularization method builds the model "globally", while the gradient boosting machine builds the model "greedily". Although the regularization method has better prediction performance in linear models, the greedy nature of the gradient boosting machine makes it more suitable for variable selection, especially when there are high correlations among the covariates in the model. In addition, the computational efficiency of the gradient boosting machine extends is application to nonlinear models.

We then implemented the gradient boosting machine on a nonlinear logistic regression model, where the regularization method fails to generate a solution. Equipped with 10-fold cross validation, we were able to choose a nearly optimal number of iterations, and the resulting model has satisfactory prediction performance. This example verifies the effectiveness of the gradient boosting machine on fitting nonlinear additive models.

In some applications, part of the output y's of the training samples are censored. In this case, the problem of fitting the model is coupled with imputing the incomplete data. Regularization methods try to estimate the regression coefficients by iteratively solve a penalized optimization problem with updated imputation. However, this

class of methods suffer from high computational cost. In some cases convergence is not guaranteed as well. It turns out that fitting a model with censored data can be treated as an additive modeling problem with infinite-dimensional nonlinear parameters. Another contribution of this thesis is a two-stage algorithm that imputes the censored data and fits an accurate model. In the first stage we use extended the L_2 boosting algorithm (PGA) to de-couple the process of imputing the residual and selecting the next variable. This procedure successfully generates a good estimate of the censored outcomes as well as the distribution of the errors. Then in the next stage we use OGA on the imputed data to rebuild the model, and remove irrelevant variables. Compared to the iterative regularization method, our two-stage algorithm has lower computational cost, and a more accurate resulting model.

Bibliography

[1] Allen, D. M. (1974). The relationship between variable selection and data augmentation and a method for prediction, *Technometrics* **16(1)**, 125-127.

[2] Boyd, S. and Vandenberhge, L. (2004). *Convex Optimization*, Cambridge University Press, Cambridge, U.K.

[3] Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984). *Classification and Regression Trees*, Wadsworth, New York.

[4] Buckley, J. and James, I. (1979). Linear regression with censored data, *Biometrika* **66**, 429-436.

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

[6] B$\acute{}$uhlmann, P. and Yu, B. (2003). Boosting with the L_2 loss: regression and classification, *Journal of the American Statistical Association* **98**, 324-339.

[7] Chen, S., Donoho, D. and Saunders, M. (2001). Atomic decomposition by basis pursuit, *SIAM Review* **43(1)** 129-159.

[8] Cox, D. R. (1972). Regression models and lifetables, *Journal of the Royal Statistical Society Series B* **34**, 187-220.

[9] Dettling, M., and B$\acute{}$uhlmann, P. (2003). Boosting for tumor classification with gene expression data, *Bioinformatics* **19** 1061-1069.

[10] Efron, B. (1983). Estimating the error rate of a prediction rule: Improvement on cross validation, *Journal of the American Statistical Association* **78(382)**, 316-331.

[11] Efron, B. (1986). How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association* **81(394)**, 461-470.

[12] Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004). Least angle regression (with discussion), *Annals of Statistics* **32**, 407-451.

[13] Fan, J. and Li, R. (2001). Variable selection via non concave penalized likelihood and its oracle properties, *Journal of the Royal Statistical Society* **70**, 849-911.

[14] Freund, Y. (1995). Boosting a weak learning algorithm by majority, *Information and Computation* **28**, 337-407.

[15] Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm, *Machine Learning: Proceedings of the Thirteenth International Conference* 148-156.

[16] Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* **55**, 119-139.

[17] Friedman, J. (1999). Stochastic gradient boosting, *Computational Statistics and Data Analysis* **38**, 367-378.

[18] Friedman, J. (2001). Greedy function approximation: a gradient boosting machine, *Annals of Statistics* **29**, 1189-1232.

[19] Friedman, J. (2008). Fast sparse regression and classification, *Technical report*, Department of Statistics, Stanford University.

[20] Friedman, J., Hastie, T. and Tibshirani, R. (2008). Regularization paths for generalized linear models via coordinate descent, *Journal of Statistical Software* **33(1)**, 1-22.

[21] Geisser, S. (1975). The predictive sample reuse method with applications, *Journal of the American Statistical Association* **70(350)**, 320-328.

[22] Gross, S. and Lai, T. (1996). Nonparametric estimation and regression analysis with left-truncated and right-censored data, *Journal of the American Statistical Association* **91(435)**, 1166-1180.

[23] Hastie, T., Tibshirani, R. & Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, Springer-Verlag, New York.

[24] Hoerl, A. E. & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems, *Technometrics* **12(1)**, 55-67.

[25] Huang, J. and Harrington, D. (2002). Penalized partial likelihood regression for right-censored data with bootstrap selection of the penalty parameter, *Biometrics* **58**, 781-791.

[26] Huang, J. Horowitz, J. and Ma, S. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models, *The Annals of Statistics* **36**(2), 587-613.

[27] Ing, C. (2007). Accumulated prediction errors, information criteria and optimal forecasting for autoregressive time series, *The Annals of Statistics* **35**, 1238-1277.

[28] Ing, C. and Lai, T. L. (2011). A stepwise regression method and consistent model selection for high-dimensional sparse linear models, *Statistica Sinica* **21(4)**, 1473-1513.

[29] Ing, C. and Wei, C. (2005). Order selection for the same-realization prediction in autoregressive processes, *The Annals of Statistics* **33**, 2423-2474.

[30] Kaplan, L. and Meier, P. (1958). Nonparametric estimation from incomplete observations, *Journals of American Statistical Association* **53**, 457-481.

[31] Koul, H., Susarla, V., and Van Ryzin, J. (1981). Regression analysis with randomly right-censored data, *The Annals of Statistics* **9**, 1276-1288.

[32] Lai, T. and Ying, Z. (1991). Large sample theory of a modified Buckley-James estimator for regression analysis with censored data, *The Annals of Statistics* **10**, 1370-1402.

[33] Lai, T. and Ying, Z. (1994). A missing information principle and M-estimators in regression analysis with censored and truncated data, *The Annals of Statistics* **22(3)**, 1222-1255.

[34] Li, H. and Luan, Y. (2005). Boosting proportional hazards models using smoothing splines, with applications to high-dimensional microarray data, *Bioinformatics* **21**, 2403-2409.

[35] Miller, R. (1976). Least squares with censored data, *Biometrika* **63**, 449-464.

[36] Ritov, Y. (1990). Estimation in a linear regression model with censored data, *The Annals of Statistics* **18**, 303-328.

[37] Schapire, R. (1990). The strength of weak learnability, *Machine Learning* **5**, 197-227.

[38] Schwarz, G. (1978). Estimating the dimension of a model, *The Annals of Statistics* **14**, 461-464.

[39] Shao, J. (1993). Linear model selection by cross-validation, *Journal of the American Statistical Association* **88(422)**, 486-494.

[40] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions, *Journal of the Royal Statistical Society* **36**, 111-147.

[41] Tempyakov, V. (2000). Weak greedy algorithms, *Advances in Computational Mathematics* **12**, 213-227.

[42] Tibshirani, R. J. (1996). Regression shrinkage and selection via the Lasso, *Journal of the Royal Statistical Society*, Series **B 58(1)**, 267-288.

[43] Tropp, J. (2004). Greed is good: algorithmic results for sparse approximation, *IEEE Transactions on Information Theory* **50**, 2231-2242.

[44] Tropp, J., Anna, C. and Gilbert, C. (2007). Signal recovery from random measurements via orthogonal match pursuit, *IEEE Transactions on Information Theory* **53**, 4655-4666.

[45] Wang, S., Nan, B., Zhu, J. and Beer, D. (2008). Doubly penalized Buckley-James method for survival data with high-dimensional covariates, *Biometrics* **64**, 132-140.

[46] Wasserman, L. and Roeder, K. (2009). High-dimension variable selection, *The Annals of Statistics* **37**, 2179-2201.

[47] Wei, L. (1992). The accelerated failure time model: a useful alternative to the Cox regression model in survival analysis. *Statistics in Medicine* **11**, 1871-1879.

[48] Wei, L., Ying, Z. and Lin, D. (1990). Linear regression analysis of censored survival data based on rank tests, *Biometrika* **77**, 845-851.

[49] Yang, Y. (1996). Consistency of cross validation for comparing regression procedures. *The Annals of Statistics* **35(6)**, 2450-2473.

[50] Zhang, P. (1993). Model selection via multifold cross validation. *The Annals of Statistics* **21(1)**, 299-313.

[51] Zou, H. & Hastie, T. (2005). Regularization and variable selection via the elastic net, *Journal of the Royal Statistical Society*, Series **B 58(1)**, 310-320.