THE UNIVERSITY OF CALGARY

Stochastic Boosting: An Application to Canadian Small Business Data

by

Shali Zhang

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS AND STATISTICS

CALGARY, ALBERTA

September, 2005

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Stochastic Boosting: An Application to Canadian Small Business Data" submitted by Shali Zhang in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Supervisor, Dr. P.F. Ehlers
Department of Mathematics and Statistics

Dr. M.D. Burke
Department of Mathematics and Statistics

Dr. E.G. Enns
Department of Mathematics and Statistics

Dr. W.W. Zwirner
Department of Applied Psychology

Sept 20/05

Date

Abstract

Combining multiple-learners into a strong one is a topic discussed by many researchers in recent years. Stochastic boosting is a new method developed recently that has shown great promise in this domain. Stochastic boosting takes random samples without replacement and generate multiple weak learners based on these subsamples. After a number of iterations, boosting combines the weak learners into a stronger one. Compared to other similar learning methods, boosting seems to have advantages in error reduction and computational load. In this thesis, I will apply stochastic boosting to a Canadian small business financing data set.

Acknowledgements

I would like to thank my supervisor, Dr. Peter F. Ehlers. He introduced me into this topic and gave valuable comments in my research. Under his instruction, I gained a lot in completing this thesis. Moreover, I would like to express my appreciation for the support from my family and, in particular, from my husband Wu.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Let us begin with a hypothetical example. The 2010 Winter Olympic Games will be held in Vancouver, B.C. With finite resources for training athletes and supporting their participation in the games, the Canadian speed skating coaching committee must select a small number of speed skaters from the pool of available skaters.

Formally, we can consider the committee's problem as one of classifying the candidates into two groups: (a) good chance of winning a medal, (b) less chance of winning a medal.

The selections made by the committee will be based on a variety of criteria: How did the candidate perform in previous competitions? Can he/she usually give reliable performance? Is he/she an experienced athlete able to tolerate intensive training and pressure? and so on.

But how should each criterion contribute to the final decision? One answer is to use historical data on past speed skaters to determine how each such criterion contributed to their (known) success. The committee could classify all past skaters into two groups: (a) excellent (category A) and (b) good (category B). Then they can try to determine classification rules based on the historical data and apply these rules to the current pool of skaters in order to classify them as excellent (likely to win a medal) or good (less likely to win a medal).

The many criteria will result in many separate decision rules. How can these be combined into a single prediction rule? The Classification and Regression Tree

(CART) model can be used. Starting with the group of all (past) skaters, we can split the group into two parts according to one criterion and then split each subgroup further into smaller but more well-defined groups according further criteria. We can keep splitting the data group until we arrive at a decision as to group membership. For example, we may split all the candidates into two parts based on their age: people older than 22 go into one part while the others go into the second part. This procedure generates a binary tree which partitions predictor space into disjoint regions R_m.

Consider a candidate skater whose predictors give a point in R_m. If more than half of the past skaters with predictor points in R_m are in category A, then the candidate will be classified as category A. The proportion of past skaters in R_m of category A is

$$\hat{p}_m = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = A)$$

where x_i and y_i represent the various predictors and the group membership of the i^{th} skater, respectively, and $I()$ is the indicator function. Here $\hat{p}_m$ is an estimate for the probability that a skater with predictors x in R_m belongs to category A.

The key part of this procedure is to determine the regions, i.e. the binary splits for the predictors. This is achieved by minimizing a loss function such as, for example, the misclassification error.

The CART model is easy to construct and interpret. However, it lacks stability which can lead to high variance and overfitting problems. Can the CART model be improved? In this thesis we discuss two methods that attempt to improve the CART model.

The first method is the *bagging* model which applies bootstrap averaging to the CART model. Bootstrap sampling is used on the original data to generate subsamples. In our example, if we have a total of 50 past skaters, we randomly subsample with replacement to obtain a new sample of size 50. Then we can generate m such bootstrap subsamples, each containing 50 observations. Each subsample will generate one binary tree and thus one prediction rule. For a given set of predictors x, this gives an estimate for the probability of category A as indicated above (see Figure 1.1). Bagging then averages these estimates. We can think of bagging as combining many weak classification rules into one stronger rule.

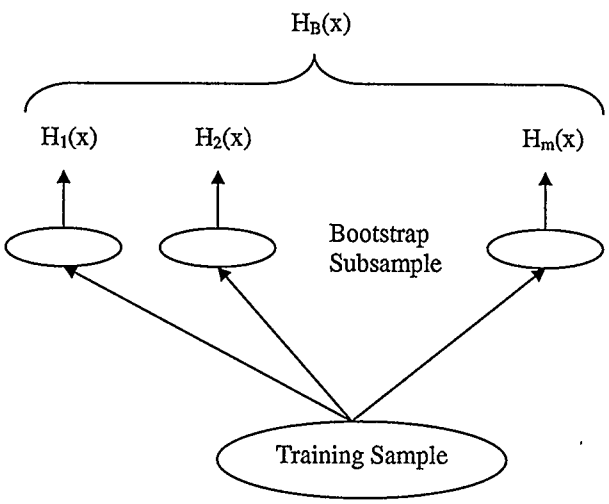

Figure 1.1: Bagging procedure.

Bagging usually gives smaller prediction error and lower variance than the CART model. However, it is much more computationally intensive.

It has become customary to use the terminology "learner" to replace "rule". We say that bagging combines many weak learners into one more powerful classifier.

This thesis is mainly concerned with *boosting*, a second method of combining many weak learners into a more powerful and efficient single rule. Boosting combines those weak learners with determinism, instead of purely averaging them as in bagging. In boosting, we also generate subsamples and each subsample also generates one weak learner, but the sampling procedure is different. In boosting, we sample without replacement to prevent the overfitting problem. Instead, subsamples are generated iteratively, with different weights given to the various data points (see Figure 1.2). Briefly, when boosting successively processes the learners, it weights the observations as follows: if an observation is not well predicted at one iteration of the process, then that observation is more weighted at the next iteration. As the iteration proceeds, subsamples will contain more useful observations because of the weights. When we stop after a suitable number of iterations, boosting will generate an optimal learner which is better than any one before.

To combine weak learners into a more powerful one was first introduced by Kearns and Vazirani [18]. Breiman [2] developed the bagging predictor in 1994. Later, Friedman [12] developed a much more efficient boosting algorithm.

In this thesis, we will focus mainly on how boosting works and how it applies to real data.

$$H(x) = sign[\sum_{m=1}^{M=1} \alpha_m H_m(x)]$$

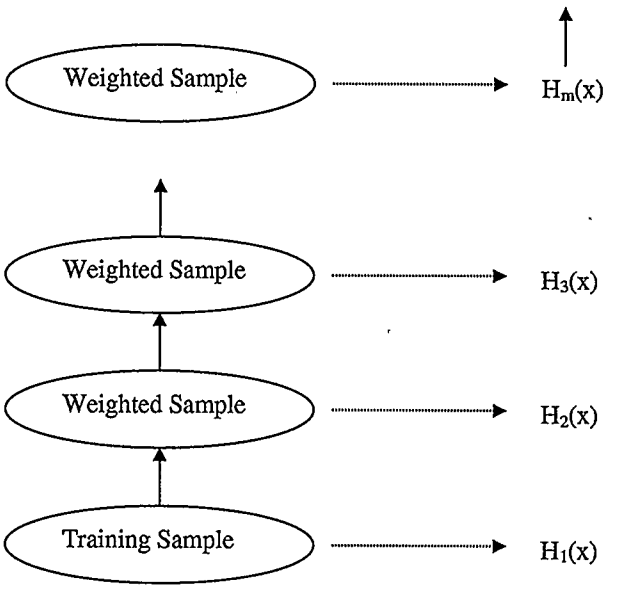

Figure 1.2: Boosting procedure.

Chapter 2 presents a short review of the bagging predictor. Chapter 3 covers the technical details of the boosting algorithm. Chapter 4 covers "regularization", i.e. improving the model by balancing different parameters. In Chapter 5 we present the results of a simulation study and in Chapter 6 we present an application to real data. Conclusions and directions for further study are given in Chapter 7.

Chapter 2

Bagging Algorithm

Bagging ("bootstrap aggregating") is a prediction procedure in which we generate many versions of a weak learner and combine them into a stronger one. It was created by Leo Breiman in 1994 [2] and it works very well in improving on the CART model when that model exhibits instability, i.e. is sensitive to small changes in the training set. In such cases bagging can provide error and variance reduction.

The bagging procedure is simple. It is based on the idea of pooling information from several samples. Suppose that we have a training data set $\{y_n, x_n\}_1^N$, where y is the response, which can be numerical or categorical, and x is the set of independent variables. CART (or some other method) provides a function $h(x)$ (the base function) which maps each x into a corresponding predicted value $y = h(x)$. If we had several training samples from the same distribution of x's and y's, each training sample giving a predictor h, then we could pool these predictors to attempt to obtain a single better predictor. For a numerical response, we can use the average of the set of base functions, i.e. we define the estimator $h_{ag}(x) = E_{\mathcal{P}}h(x)$, where $\mathcal{P}$ is the distribution from which the samples are obtained.

Of course, normally we do not have several training samples. Bagging obtains multiple training samples by forming bootstrap samples from the given training sample. Each bootstrap replication provides one new version of the base function. If Z is the training sample with N observations, then we take bootstrap samples of size N from Z, with replacement, to obtain a sequence of bootstrap training sets

$\{S(Z^1), S(Z^2), \ldots, S(Z^k)\}$.

All of them contain N observations; moreover, because of replacement, there could happen that a certain observation appears several times in one or even some bootstrap samples, or there could be some certain observations never appear in any samples. It leads to overlapping problem in bagging which we will explain more latter.

After generating bootstrap samples, each of them has the same base function as their "mother" training sample $\{h_{B1}(x), h_{B2}(x), ..., h_{Bk}(x) = h(x)\}$. And all of them would be combine into a powful one:

$$h_B(x) = E_{x,y}(x, h_i(x)) \tag{2.1}$$

It is the aggregated prediction based on all the weak learners and called "bootstrap aggregating". It is called "bagging".

Breiman [2] said bagging works well for unstable procedures. Actually in our simulation study, bagging does show its priority in variance reduction when compared with boosting. But how does bagging work?

y, x are the random variables taken from training sample, so the average prediction error ϵ in $f(x)$ is

$$\epsilon = E(y - h(x))^2 \tag{2.2}$$

While the error in aggregated predictor is

$$\epsilon_B = E(y - h_B(x))^2 \tag{2.3}$$

Readjust: $\epsilon = E(y - h_B(x) + h_B(x) - h(x))^2 = E(y - h_B(x))^2 + 2E((y - h_B(x))(h_B(x) - h(x))) + E(h_B(x) - h(x))^2$ Obviously, we can have $\epsilon \geq E(y - h_B(x))^2$. Thus, ϵ_B has lower mean-squared prediction error than ϵ, but how much lower depends on how unequal the two sides if:

$$[Eh(x)]^2 \leq Eh^2(x) \tag{2.4}$$

We will show the comparison between boosting and bagging in the last two chapters. However, bagging creates a new method which can combine weak learners into a strong one. Its error reduction and variance restriction are much better than a single tree. But overlapping problem in data sampling prevents bagging from better performance in error reduction. Because it works like a blind when picking samples; it just keeps taking data and returning it without any purpose so that it could not catch all the data or absorb all the information from them. Boosting is another bagging algorithm with eyes. It takes subsamples randomly without replacement. But it can "see" the contribution of each observation and weight them. Therefore, according to the weights, boosting knows whether to take or ignore the corresponding data.

Breiman also tried to explain the difference between bagging and boosting in [4, 5]. However, he addressed this issue in terms of the bias and variance decomposition of the error. But we think "boosting-by-reweighting" is the key point between them.

Chapter 3

Boosting Algorithm

Boosting is a model averaging technique that combines the classification or regression predictions of several models. It has been found, over the past decade, to be one of the most effective methods in improving accuracy in statistical learning problems, particularly in terms of reducing the error in a training data set and in preventing overfitting.

It originally stemmed from a question raised by Kearns and Valiant [17] regarding the equivalence of strong and weak learnability. Schapire [26] settled the question in the affirmative by proving his "Strength of Weak Learnability" theorem and devising the first simple boosting algorithm.

Freund [8] improved Schapire's algorithm by combining many weak learners simultaneously. Breiman [2] tackled the problem of improving a weak learner with his "bagging predictor" which randomly generates weak classifiers by bootstrapping and then combines them into a single more powerful one. However, bagging does not give "boosted" performance on error reduction.

In 1999, Freund and Schapire [11] devised the adaptive boosting algorithm "AdaBoost.M1" which was a relatively mature algorithm of boosting and solved many of the practical difficulties of earlier boosting attempts.

In this chapter, we present details of the boosting procedure first for the 2-state classification problem and then for the regression problem. Our focus is on tree models.

3.1 Two-State Classification

This section follows the treatment in Hastie *et al.* [16, chapter 10] and in the original papers cited.

Consider a two-state classification problem, where the response Y belongs to $\{-1, 1\}$ and the predictor vector X is mapped into Y by the classifier $H(x)$, called the "base function". The error rate in the training set is

$$\overline{err} = \frac{1}{N} \sum_{1}^{N} I[y_i \neq H(x_i)]$$

Assuming that the classifier H is a weak learner, we attempt to boost it into a strong learner. Boosting repeatedly applies modified versions of the weak learner to the training data, using different weights for the observations as the algorithm proceeds. This process results in a sequence of weak learners. These are then combined as a linear combination with coefficients determined by the boosting algorithm. The resulting classifier has been found to be much better than the original weak learner.

The algorithm for boosting a two-class weak learner is known as Discrete AdaBoost. It is shown in Table 3.1 [see 16, p. 301].

In the beginning, all the observations are given the same weight $1/N$; as iterations proceed, the weights are modified and the current classifier is applied to the weighted observations. If some observations are misclassified by $H_{m-1}(x)$, they will be more heavily weighted in $H_m(x)$; in other words, the harder they are to classify, the more weight they earn. Thus, as iteration proceeds, observations that are difficult to be classified correctly receive ever-increasing influence. Each successive classifier is thereby forced to concentrate more on those training observations that are misclassified by the previous classifier in the sequence.

The weights are used in minimizing a weighted loss function.

Algorithm: Discrete AdaBoost

---

1. set the initial weights as $w_i = \frac{1}{N}$
2. For $m = 1$ to M:
   (a) fit the classifer $H_m(x) \in \{-1, 1\}$ with weights w_i
   (b) let $err_m = \frac{\sum_1^N w_i I(y_i \neq H_m(x_i))}{\sum_1^N w_i}$
   (c) let $\beta_m = \frac{1}{2}\log((1 - err_m)/err_m)$
   (d) update the weights: $w_i \leftarrow w_i \exp[2\beta_m I(y \neq H_m(x_i))]$
3. Output $H(x) = \text{sign}\left[\sum_{m=1}^M \beta_m H_m(x)\right]$

Table 3.1: Algorithm for Discrete AdaBoost

The output from the algorithm is the boosted classifier $H(x) = \text{sign}\left[\sum_1^M \beta_m H_m(x)\right]$ which involves a linear combination of base functions $H_m(x)$, each with value in $\{-1, 1\}$. $H(x)$ may be thought of as a weighted majority vote of the M base functions. This boosted classifier can "boost" error reduction dramatically.

We will now show that the Discrete AdaBoost algorithm fits an additive model with exponential loss function by a procedure known as *forward stagewise* modeling [14].

An additive model is of the form $f(x) = \sum \beta_m\, b(x; \gamma_m)$ for some (usually simple) functions $b()$ and parameters γ. (For tree models, γ determines the split variables and the split points.)

An additive model may be fit by minimizing a loss function over the training data. If $L(y, f)$ is the loss function for model f and response y, then we seek to minimize $\sum_i L(y_i, f(x_i))$ over the parameters in f:

$$\min_{\beta,\gamma} \left\{ \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m \, b(x_i; \gamma_m) \right) \right\}$$

Forward stagewise modeling seeks an approximate solution to the minimization problem by forming the m^{th} solution as a sum of the previous solution and a new optimal term, but without adjusting any previously obtained parameter values:

$$f_m(x) = f_{m-1}(x) + \beta_m \, b(x; \gamma_m)$$

The optimal parameters are obtained from

$$(\beta_m, \gamma_m) = \underset{\beta,\gamma}{\operatorname{argmin}} \left[\sum_{i=1}^{N} L(y_i, \, f_{m-1}(x_i) + \beta \, b(x_i; \gamma)) \right]$$

Consider now the 2-state classification problem with classifiers H_m and exponential loss function

$$L(y, f(x)) = \exp(-y f(x))$$

where $f(x) = \sum \beta_m H_m(x)$ and we must solve for H_m and β_m so as to minimize total loss. The loss is large if y and $f(x)$ have opposite signs and small if they have the same sign.

Let $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$. Then

$$\begin{aligned}
(\beta_m, H_m) &= \underset{\beta,H}{\operatorname{argmin}} \sum_{i=1}^{N} \exp\left[-y_i(f_{m-1}(x_i) + \beta H(x_i))\right] \\
&= \underset{\beta,H}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \exp(-y_i \beta H(x_i)) \\
&= \underset{\beta,H}{\operatorname{argmin}} \left[e^{-\beta} \sum_{y_i = H(x_i)} w_i^{(m)} + e^{\beta} \sum_{y_i \neq H(x_i)} w_i^{(m)} \right] \\
&= \underset{\beta,H}{\operatorname{argmin}} \left[\left(e^{\beta} - e^{-\beta}\right) \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq H(x_i)) + e^{-\beta} \sum_{i=1}^{N} w_i^{(m)} \right]
\end{aligned}$$

Assuming that β is positive, H_m is given by

$$H_m = \underset{H}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq H(x_i))$$

Then, solving for β_m gives

$$\beta_m = \frac{1}{2} \log \frac{1 - err_m}{err_m}$$

where

$$err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq H(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}$$

The forward stagewise approximation

$$f_m(x) = f_{m-1}(x) + \beta_m H_m(x)$$

gives new weights for the next iteration

$$\begin{aligned}
w_i^{(m+1)} &= \exp\left[-y_i f_m(x_i)\right] \\
&= \exp\left[-y_i(f_{m-1}(x_i) + \beta_m H_m(x_i))\right] \\
&= w_i^{(m)} \times \exp[2\beta_m I(y_i \neq H_m(x_i))] \times \exp(-\beta_m) \\
&\propto w_i^{(m)} \times \exp[2\beta_m I(y_i \neq H_m(x_i))]
\end{aligned}$$

which is the weights updating formula given in the Discrete AdaBoost algorithm. (In the 3rd line, we have used the fact that when both y_i and $H_m \in \{-1, 1\}$, then $y_i H_m(x_i) = 1 - 2I(y_i \neq H_m(x_i).)$

Thus we have shown that the Discrete AdaBoost algorithm fits an additive model by the forward stagewise approximation, using an exponential loss function.

Other commonly considered classification loss functions are

- misclassification: $L(y, f) = I(y \neq \text{sign}(f)) = I(yf < 0)$
- binomial deviance: $L(y, f) = \log(1 + \exp(-2yf)$
- squared error: $L(y, f) = (y - f)^2$

All four loss functions are shown in Figure 3.1 where each function is scaled so that it passes through the point $(0, 1)$. The functions are plotted as functions of the "margin" yf.

With a classification rule of $H(x) = \text{sign}(f(x))$, positive margin indicates correct classification while negative margin values indicate misclassification. Negative margins should be penalized.

Hastie *et al.* [16] point out that the exponential and the binomial deviance loss functions can be considered as monotone continuous approximations to misclassification loss. They also note that binomial deviance is more robust than exponential loss, especially for noisy data, because it gives less influence to observations with large negative margin. This spreads the influence more evenly among all the data.

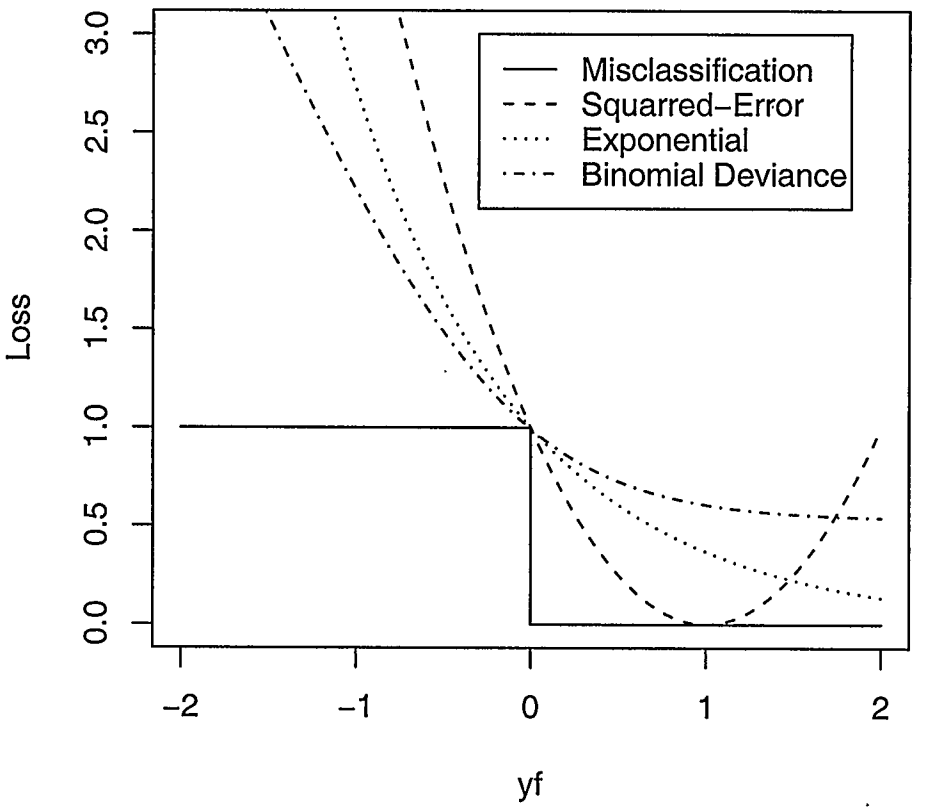

Figure 3.1: Loss functions for two-state classification.

3.2 Boosting Trees

3.2.1 Classification Trees

In a tree model, we partition the predictor space into disjoint regions R_j, represented by the terminal nodes of the tree. Region R_j corresponds to a value γ_j and the classification rule is that $f(x) = \gamma_j$ if $x \in R_j$. For a two-state model, $\gamma_j \in \{-1, 1\}$.

Tree models are obviously well interpretable [6]. Under the decision rule, a binary tree model separates the data into two primary regions; each resulting node can then be split into two sub-regions, and so on.

The binary tree model can be written as a linear combination of indicator functions:

$$T(x; \Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$$

where $\Theta = \{R_j, \gamma_j\}_1^J$ is the set of parameters in the model which are chosen to minimize total loss:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_j \sum_{x_i \in R_j} L(y_i, \gamma_i)$$

We build up a boosted classifier as a sum of tree models

$$f_M(x) = \sum_{m=1}^{M} \beta_m T(x; \Theta_m)$$

In boosting trees, a tree-based classifier is employed by AdaBoost and it generates the final classifier which usually proves to be much better than the single tree. Breiman [5] called AdaBoost with tree model the "best off-the-shelf classifier in the world".

Let us carry out the forward stagewise procedure again. At each step,

$$\hat{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta_m T(x_i, \Theta_m)) \tag{3.1}$$

must be solved for the next tree region given current model $f_{m-1}(x)$. For exponential loss, this becomes

$$\hat{\Theta}_m = \underset{\Theta_m}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \exp\left[-y_i \beta_m T(x_i; \Theta_m)\right] \tag{3.2}$$

3.2.2 Regression Trees

We can apply the AdaBoost algorithm to the case of a numerical response variable. For binary tree models, the disjoint regions of predictor space lead to terminal tree nodes which carry the numerical response. A special case is when the node values are the estimated probabilities $P(Y = 1|x)$ for the two-state classification problem discussed above.

The results stated for classification trees essentially carry over to regression trees. The algorithm again uses forward stagewise modeling. The final estimate is a sum of estimates. Commonly used loss functions are (see Figure 3.2)

- squared error: $L(y, f) = (y - f)^2$
- absolute error: $L(y, f) = |y - f|$

Comparing squared error loss and absolute error loss, we can see that for small margin $y - f(x)$, squared error can give lower loss than absolute error. However, for larger margin, absolute error does better.

Another useful loss function is Huber loss which combines the desirable properties of the squared error and absolute error loss functions. Huber loss is not used in this thesis.

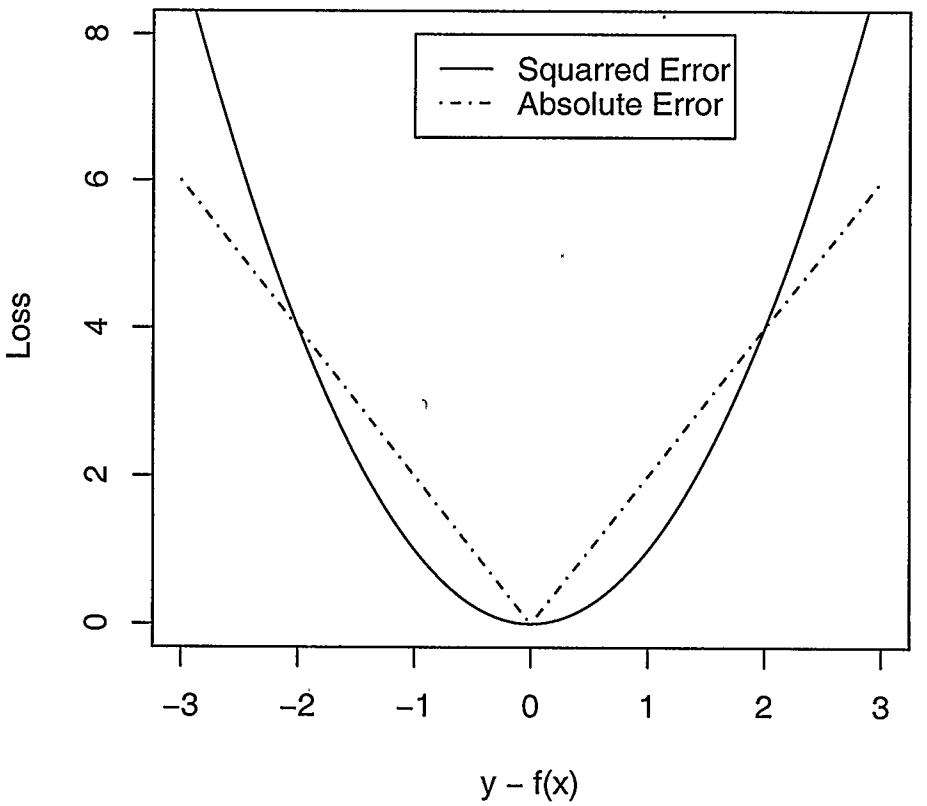

Figure 3.2: Loss functions for Regression

3.2.3 Numerical Details

To solve Equation 3.1, the *gradient boosting* method is used. This is essentially a steepest descent method applied to the loss function with respect to the fitting functions. Let

$$L(\mathbf{f}) = \sum_{i=1}^{N} L(y_i, f(x_i))$$

where $\mathbf{f}$ is the vector of fitting functions at the data points, $\mathbf{f} = \{f(x_1), f(x_2), \cdots, f(x_N)\}$. Minimizing loss is then given by

$$\mathbf{f} = \underset{\mathbf{f}}{\operatorname{argmin}}\, L(\mathbf{f})$$

which can be done stepwise as

$$\mathbf{f}_M = \sum_{m=0}^{M} \mathbf{f}_m$$

with recursion formula

$$\mathbf{f}_m = \mathbf{f}_{m-1} - \rho_m \mathbf{g}_m.$$

Here $\mathbf{g}_m$ is the gradient of the loss function with i^{th} component

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$$

evaluated at $\mathbf{f}_{m-1}$ and ρ_m is the optimal step size given by

$$\rho_m = \underset{\rho}{\operatorname{argmin}}\, L(\mathbf{f}_{m-1} - \rho \mathbf{g}_m)$$

The gradient boosting algorithm applies the steepest descent procedure subject to the constraint that the prediction functions $f_m(x)$ are the predictions of a tree of size J_m. For details of the algorithm, see Hastie *et al.* [16, p. 322].

3.3 Stochastic Boosting

With the "bagging" procedure, Breiman [2] injected randomness into the function estimating procedure. The bagging model takes random subsamples from the training sample, *with replacement* and of size equal to that of the training sample. Each subsample produces a predicted response and these predictions are aggregated in order to achieve an improved prediction.

In contrast to bagging, ordinary boosting does not resample the training data. However, Friedman [13] introduced the idea of *subsampling* from the training data. He defines a new parameter, the subsampling fraction f, which is the fraction of the training data that is sampled at each iteration of the procedure. Thus, at each iteration, a random sample of size $\hat{N} = q \times N$ is obtained from the training data. Then the current model is based on this subsample. This modified boosting procedure is called "stochastic boosting".

If $q < 1$, then the subsample drawn at each iteration will be different. This will help to prevent overfitting in addition to computational savings.

How to choose the value of q? The smaller q is, the more randomness the procedure will have. Up to a point, this is desirable. But if the subsample is too small, then it will contain insufficient information to build useful models.

Friedman [13] demonstrates that the improvement is obvious when $0.5 \leq q \leq 0.8$.

The degree of improvement of stochastic boosting depends on the case at hand: the sample size; the target function; whether regression or classification. Chapter 5 will illustrate the performance of stochastic boosting under various conditions.

Chapter 4

Regularization

In the prediction problem, we would like to balance between the model fitting and overfitting problems. Hastie *et al.* [16] divide it into two parts:

1. Model Selection: estimating the performance of different models in order to choose the (approximately) optimal one.

2. Model Assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

To solve these problems, we divide the data set into two sets. One is the training set which is used to build the model; the other one is the testing set which is used to assess the generalization error for the final model. If we have a large data set, we may add another part, called the "validation set", which is used to estimate the prediction error for model selection. By dividing the whole data set into such two or three parts, we connect the model-building process to model prediction.

But it is not reasonable to use a fixed fraction for each of these three parts since this depends on the signal-to-noise ratio in the data and the training set size. Hastie *et al.* [16] suggest that a typical split might be 50% for the training sample and 25% each for validation and testing samples. But we are often faced with an insufficient data set that is too small to be split into three parts. Breiman [2] keeps 80% as the training data and uses 20% as the testing data in a simulation study while he would use 20% as the training data and 80% as the testing data in a real

data study. However, the choice of the proportion between the training sample and testing sample is somewhat subjective.

In the gbm procedure, we will first manually split the full data set into training and testing sets. Then we will divide the training set into training set and validation set. According to the training error and validation error, we can adjust some parameters of the model so that we can get an optimal model which not only fits the data very well but also avoids the overfitting problem. There are three parameters which can be adjusted to achieve "regularization".

4.1 Tree Size

Boosting is a method of combining weak learners into a strong one. The boosted tree model is a sum of trees:

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

Thus we consider a tree as our weak learner $H(x; \alpha)$. First of all, a large tree is generated. However, it will follow the training set so closely that it will not generalize well in prediction with the test data. Hence the tree is pruned to the optimal number J of terminal nodes. At each iteration, we will then restrict all trees to be of the same size. If J is too large, predictive performance is poor and computations are increased. However, if J is too small, then the model will be too coarse to fit the data well and again we will have poor predictive performance. Thus J becomes the primary metaparameter of these weak learners. "The best choice for its value depends most strongly on the nature of the target function, namely, the highest order of the dominant interactions among the variables" [12].

Here the target function is:

$$\eta(x) = \underset{f}{\operatorname{argmin}} E_{Y|x} L(Y, f(x))$$

One property of $\eta(x)$ is the degree to which the coordinate variables $x = (x_1, x_2, x_3, \ldots, x_p)$ interact with one another. We can write the target function as an ANOVA expansion:

$$\eta(x) = \sum_j \eta_j(x_j) + \sum_{jk} \eta_{jk}(x_j, x_k) + \sum_{jkl} \eta_{jkl}(x_j, x_k, x_l) + \ldots$$

The first sum is called the "main effects" component of $\eta(x)$. It consists of a sum of functions each depending on only one component of x. The second sum consists of functions which represent the two-variable "interaction effects". The third summation represents third order interaction and so on.

The highest interaction order possible is limited by the number of input variables. However, in practice, the target function is generally determined by lower order interactions.

In boosting trees, the interaction level is controlled by the tree size J. The highest order interaction cannot be larger than $J - 1$. In the gbm package, the tree size is controlled by the `interaction.depth` argument. For a single split tree ($J = 2$), `interaction.depth` is set equal to 1. This will produce the main effect. $J = 3$ would give the two-variable interaction; and so on.

To demonstrate the effect of tree size on the training and validation error, we use the simulated data in the example provided with the gbm package. The data consist of 1000 observations. Squared error loss is used. Half of the observations are regarded as the training set and the rest form the validation set. The algorithm

is run for 300 iterations. In Figure 4.1 we give boxplots of the training error and validation error for $J \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. From the left plot, we see that, as the value of J increases, the median training error decreases.

The right plot in Figure 4.1 shows the validation error. Unlike for training error, validation error does not necessarily decrease with increases in J. It is not difficult to explain this behaviour. In the training set, we focus on how well the model fits the data. Thus the larger J value representing a more complex tree model can capture the pattern in the training sample more accurately. But when the model gives too good a fit on the training sample, it will lose predictive power in the testing sample, because it follows the training sample so closely. As a result, it will give poor performance on validation or testing data.

An optimal model should balance between model-fitting and over-fitting. Although small trees with $J = 2$ are insufficient, $J = 10$ is unnecessary and subject to over-fitting. Generally, we consider $4 \leq J \leq 8$ to be adequate.

4.2 Iteration and Shrinkage

Besides the tree size, there are two other parameters that can be regarded as part of regularization: one is the number of iterations M; the other is the shrinkage v. In boosting, each iteration will reduce the expected risk as given by the loss function. As long as the iteration continues, the error will keep decreasing. But if training error becomes too small, the model will fit the training data too well to do future prediction in testing data. So we should find an optimal number M^* of iterations which will not only fit the training data well but also predict well. We can think of

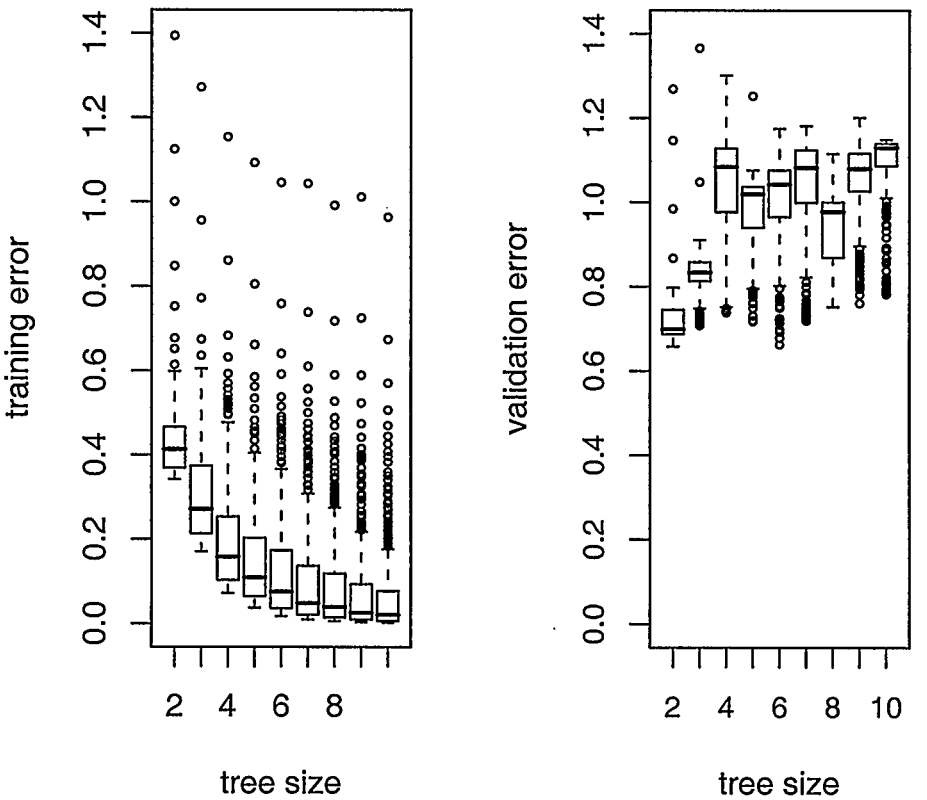

Figure 4.1: Different tree sizes based on 300 iterations.

J (tree size) as a measure of model complexity in terms of the complexity of a base learner, while M is a measure of model complexity in terms of the number of base learners included in the model.

In practice, we first find several values of M (several models) for which the training error is acceptable and then apply them on a validation sample to estimate "prediction error". From these models, we can pick one which has acceptable validation error.

Another important regularization parameter that can be tuned is the shrinkage.

Hastie *et al.* [16] add the parameter v as learning rate into the update part of the boosting procedure:

$$f_M(x) = f_{M-1}(x) + v \sum_{j=1}^{J} \widehat{\gamma}_{jm} I(x \in R_{jm})$$

We control $f_M(x)$ by scaling the contribution of each tree. Here we set $v \in (0, 1]$. The smaller v is, the less information $f_M(x)$ will get from each tree. If $v = 1$, there is no shrinkage. We can say that shrinkage slows the learning process of the model. Small values of shrinkage are preferred because they lead to improved prediction. However, they also require more computational effort.

In Figure 4.2, under the same data randomly generated in Section 4.1, we applied the shrinkage value equal to $\{1, 0.8, 0.6, 0.4, 0.2, 0.05, 0.01, 0.005\}$ with 300 iterations. In the left plot of training error reduction, we can tell $v \in \{1, 0.8, 0.6, 0.4, 0.2\}$ give lower value of training error than the other three values $\{0.005, 0.01, 0.05\}$. Moreover, when there is no shrinkage, $v = 1$, the training error is larger than that of $v = 0.8, 0.6, 0.4$. Therefore, we can see boosting model with shrinkage can give lower training error than without shrinkage. But if the value of shrinkage is too small, like

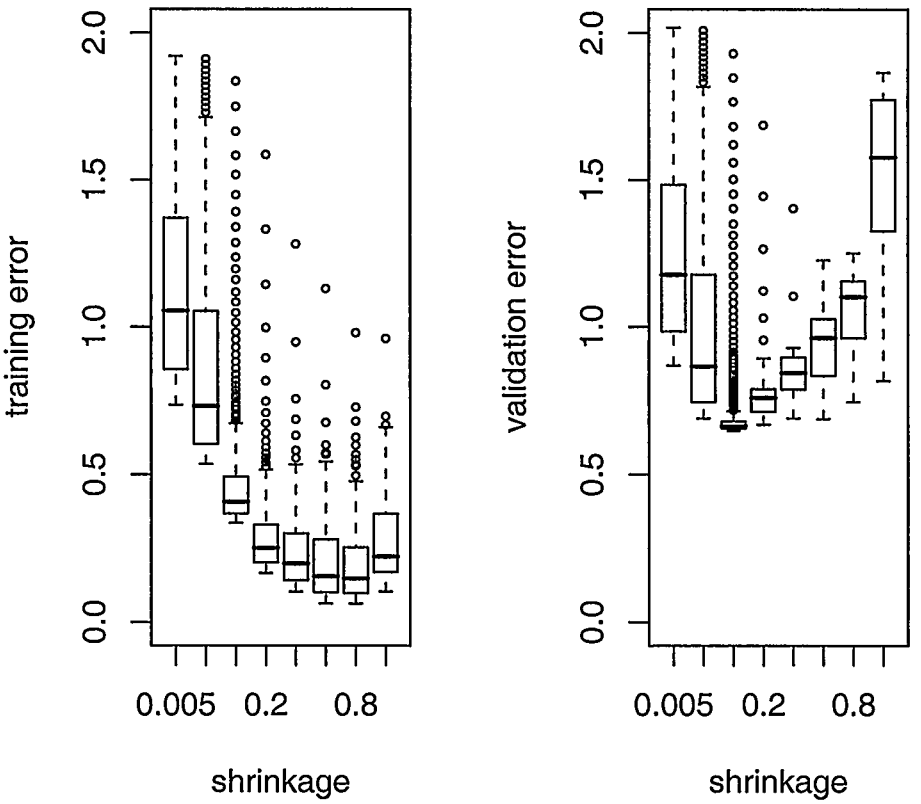

Figure 4.2: Different values of shrinkage on 300 iterations.

$0.005, 0.01, 0.05$, the situation can even worsen.

From the right plot of validation error, we can find the optimal value of shrinkage. The validation errors from $\{0.05, 0.2, 0.4, 0.6, 0.8\}$ are lower than the validation errors from $\{0.005, 0.01, 1\}$. Balancing between training error and validation error, we think the optimal value of shinkage can be 0.2.

Figure 4.3 presents the effect of shrinkage on training error for selected values of the shrinkage parameter and a range of number of iterations. For the largest three values of shrinkage, the training error decreases rapidly during the first few iterations and then continues to decrease more slowly.

When the shrinkage parameter is set at 0.005, training error also decreases with increasing number of iterations but the decrease is much slower. But if sufficient number of iterations are performed, the training error will decrease to the same level as for the other shrinkage values.

Since $v = 0.2$ is the optimal shrinkage value as found above, we plug it into our model to test the effect of the number of iterations. The number of iterations M is $\{100, 300, 500, 800, 1000, 1500, 2000\}$.

In Figure 4.4, we can see that, even though $v = 0.2$ is the optimal value of shrinkage to reduce the training error under 300 iterations, we still can reduce the training error considerably more by adding more iterations. The left plot shows that the training error keeps decreasing when the number of iterations increases; the minimum value is about $M = 1500$ or $M = 2000$. However, too many iterations can lead to overfitting.

The right plot in Figure 4.4 shows that the validation error increases when more iterations are added. To balance between training error and validation error, we

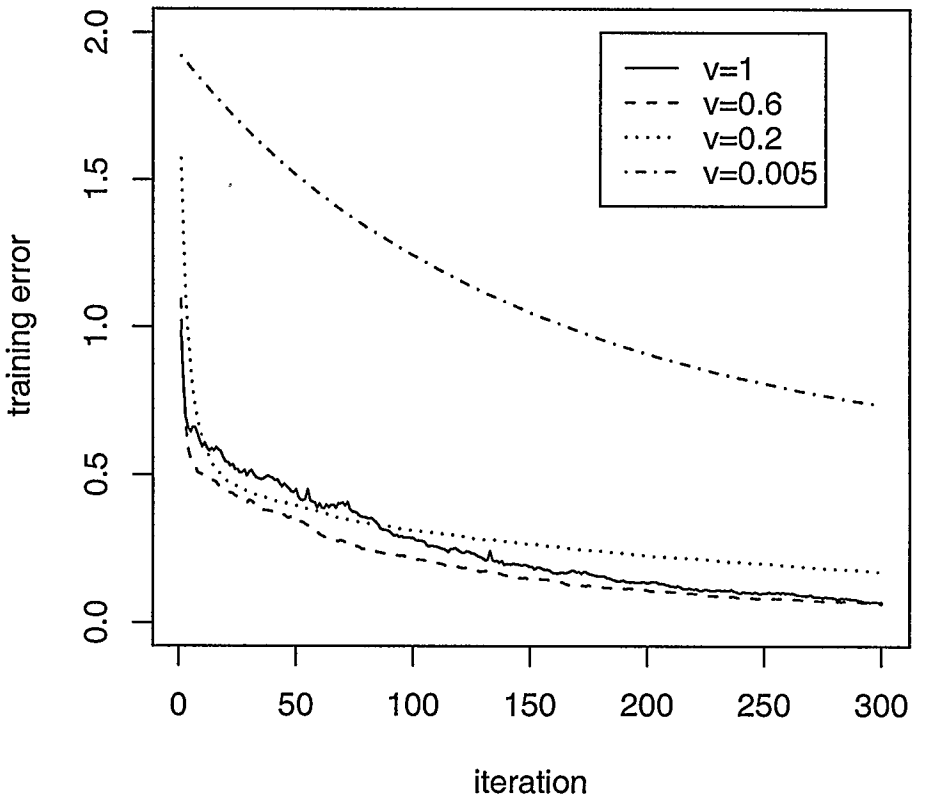

Figure 4.3: Performance of shrinkage under fixed iteration

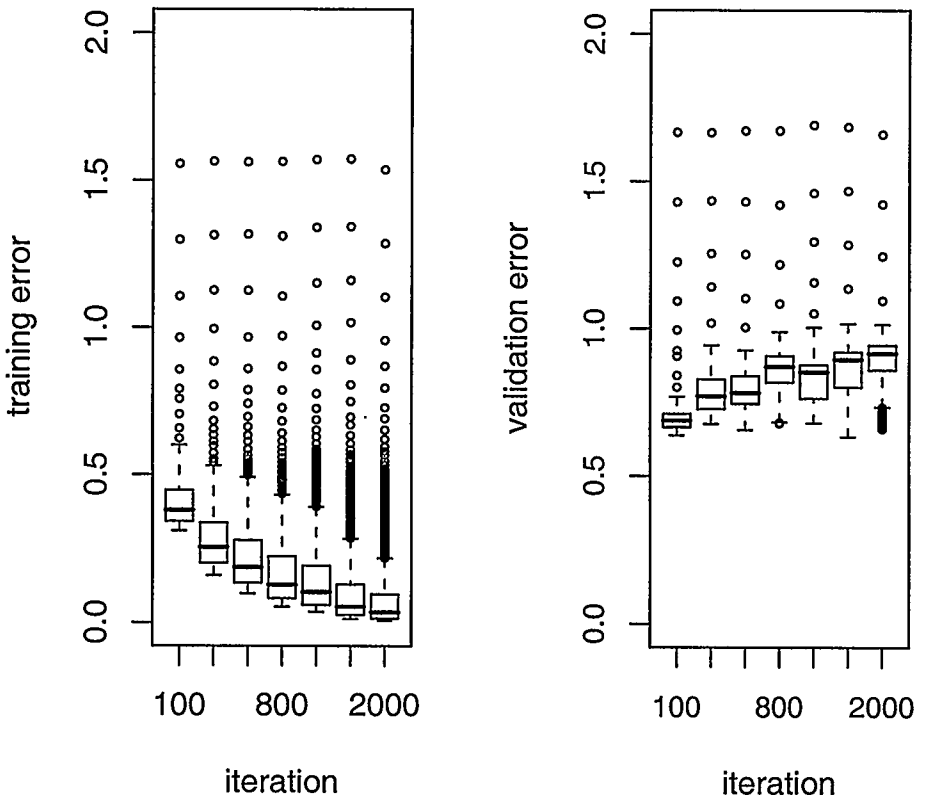

Figure 4.4: Performance of iteration under fixed shrinkage

choose $M = 500$ as the optimal value because training error decreases slightly from $M = 300$ to $M = 500$ while the validation error remains the same.

The optimal combination of tree size (J), shrinkage (v) and number of iterations (M) depends on the particular data set at hand. We use inital values based on our experience but we should then experiment and compare with other values to get the optimal combination which will reduce the training error and still prevent overfitting.

Hastie *et al.* [16] suggest that the balance between the iteration and shrinkage can yield dramatic improvements (over no shrinkage) for regression and for probability estimation. The corresponding improvements in misclassification risk are smaller but still substantial.

Chapter 5

Simulation Study

In this chapter, we present the results of a simulation study of the boosting algorithm. Essentially, we follow the simulation procedure in Friedman *et al.* [14], but we simplify the procedure slightly.

We generate both a large data set ($N = 10000$ cases) and a small data set ($N = 2000$ cases). The large set is divided into a training sample of $N_1 = 7500$ cases and a test set of $N_2 = 2500$ cases. Similarly, the small data set is divided into $N_1 = 1500$ and $N_2 = 500$ cases, respectively.

Each case will consist of the values of 10 predictor variables and one response variable. The predictor values are generated i.i.d. $N(0, 1)$. Thus, the predictors are uncorrelated. Generating the response values is more complicated because we want the response (target function) to be a very general *additive* function of the predictors. (We use a sum of 20 terms in order to provide a reasonable degree of complexity.) Let the target function be of the form

$$F^*(x) = \sum_{\ell=1}^{20} \alpha_\ell g_\ell(Z_\ell) \tag{5.1}$$

where the coefficients α_ℓ are randomly generated from the uniform distribution $U[-1, 1]$ and Z_ℓ is a randomly selected subset of size n_ℓ of the 10 predictors, i.e. $Z_\ell = \{x_{P_\ell(j)}\}_{j=1}^{n_l}$. Here, P_ℓ is a random permutation of the integers $\{1, 2, \ldots, n\}$. The size n_ℓ of each subset is taken to be random, $n_\ell = \lfloor 1.5 + r \rfloor$ with r being drawn from an exponential distribution with mean $\lambda = 2$. (If we obtain $n_\ell > 10$, then we

simply resample.) Thus, the expected length of the vector Z_ℓ is between 2 and 4.

The value of $g_\ell(Z_\ell)$ is obtained from an n_ℓ-dimensional Gaussian function

$$g_\ell(Z_\ell) = \exp\left\{-\frac{1}{2}\left[(Z_\ell - \mu_\ell)^T V_\ell (Z_\ell - \mu_\ell)\right]\right\} \tag{5.2}$$

where the components of the mean vector μ_ℓ are i.i.d. $N(0,1)$.

In order to generate V_ℓ, we first generate the n_ℓ eigenvalues of the matrix $D_\ell = \text{diag}\{d_{1\ell}, \ldots, d_{n_\ell \ell}\}$ via $\sqrt{d_{j\ell}} \sim U[a,b]$, where $a = 0.1, b = 0.2$. With U_ℓ a random orthonormal matrix, we then define $V_\ell = U_\ell D_\ell U_\ell^T$.

The above procedure generates a value for F^*. To generate the response, we then perturb $F^*(x)$ with a random error term ε, generated from a normal distribution with zero mean and variance given as follows. Let $G_\ell = \alpha_\ell g_\ell(Z_\ell)$ so that $F^*(x) = \sum G_\ell$. Then we let

$$\sigma = \frac{1}{20}\sum_{i=1}^{20} |G_\ell - \text{median}\,(G)| \times \frac{\sqrt{2\pi}}{2} \tag{5.3}$$

Then (x_i, y_i) with $y_i = F^*(x_i) + \varepsilon_i$ is the i^{th} simulated case and we generate N such cases. In the following sections, we will apply these simulated data to investigate: 1) the performance of stochastic boosting for different sampling fractions; and 2) the relative importance and partial dependence of the predictors.

5.1 Stochastic Subsampling Fraction

In Chapter 3, we discussed *stochastic* boosting theoretically. A new parameter, the subsampling fraction q, is chosen to resample from the full training set. The subsample generated at each iteration is used to fit the base learner each time. Compared to deterministic boosting, stochastic boosting resamples the weighted

observations randomly *without* replacement. This saves computational time but, more importantly, creates a final model that is more robust against overfitting. Since a new radom sample is obtained at each iteration, stochastic subsampling tends to reduce correlation between the models generated at different iterations.

There is an optimal value for the subsample fraction q between 0 and 1. Choosing $q = 1$ is equivalent to deterministic boosting — the model will lose the robustness due to randomness. On the other hand, if q is too small, there will be insufficient data for building reliable models. In our simulations, we compare results for values of q in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0\}$.

5.1.1 Regression

Figure 5.1 shows boxplots of the training error when we applied stochastic boosting to the large-sample simulated data with different choices for q.

Reading from Figure 5.1, values of q in $\{0.3, 0.4, 0.5, 0.6, 0.8\}$ give nearly identical training errors which are lower than those for q in $\{0.1, 0.2, 1.0\}$. As expected, the two extreme values result in higher training error. If $q = 1$, there is no random subsampling and predictions will suffer from overfitting. However, if the q-value is too small, only a few observations are subsampled in each iteration and they cannot build a reliable model.

Next we consider the validation error plot which evaluates the capability of prediction. From Figure 5.2, the optimal value of q can be chosen from $\{0.3, 0.4, 0.5, 0.6, 0.8\}$. Since the training errors for those five q-values are quite similar, we choose $q = 0.5$ as the optimal subsampling fraction, because it gives the smallest validation error.

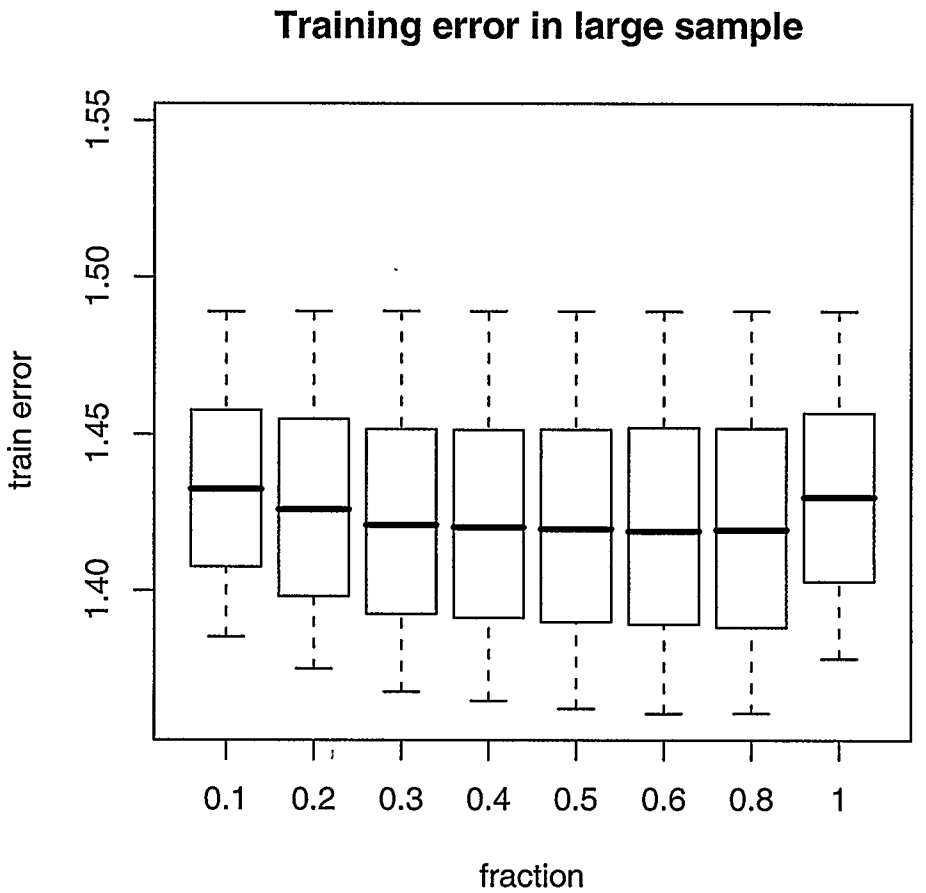

Figure 5.1: Training error under different random factors.

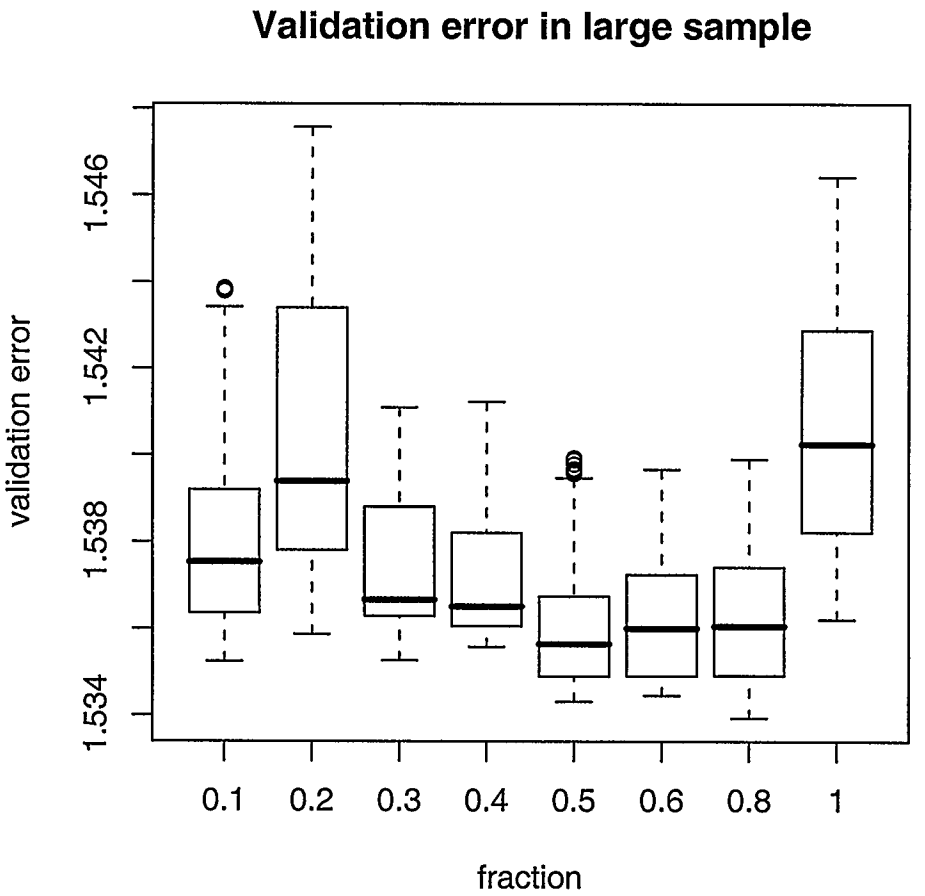

Figure 5.2: Validation error under different random factors.

For the small simulated sample ($N = 1500$), the training error is shown in Figure 5.3 which is quite similar to Figure 5.1. In the small data set, values of q in $\{0.4, 0.5, 0.6, 0.8\}$ give similar training errors which are lower than the other four. In order to find an optimal value, we look at the validation error in Figure 5.4. The lowest validation error occurs for $q = 0.4$ or $q = 0.8$. We take $q = 0.8$ as the optimal value in the small data set because its validation error distribution has less variance and slightly smaller mean than that for $q = 0.4$.

After the evaluation of training error (Figure 5.3) and validation error (Figure 5.4), we come to the model assessment stage which involves prediction with the test data. Here we use *relative absolute error* as the statistical criterion:

$$\text{rel. err}_i = \frac{|y_i - \hat{y}_i|}{\min |y_i - \hat{y}_i|} \tag{5.4}$$

Figures 5.5 and 5.6 display the distributions of relative error for the large-sample ($N_2 = 2500$) and small-sample ($N_2 = 500$) simulations, respectively, for different values of the subsampling fraction. In the large-sample case, $q = 0.6$ obviously gives widely dispersed errors, while the optimal value $q = 0.5$ (as selected in the validation stage) gives fairly consistent errors. For the small-sample case, the optimal value $q = 0.8$ gives very consistent errors, while deterministic boosting leads to highly dispersed errors.

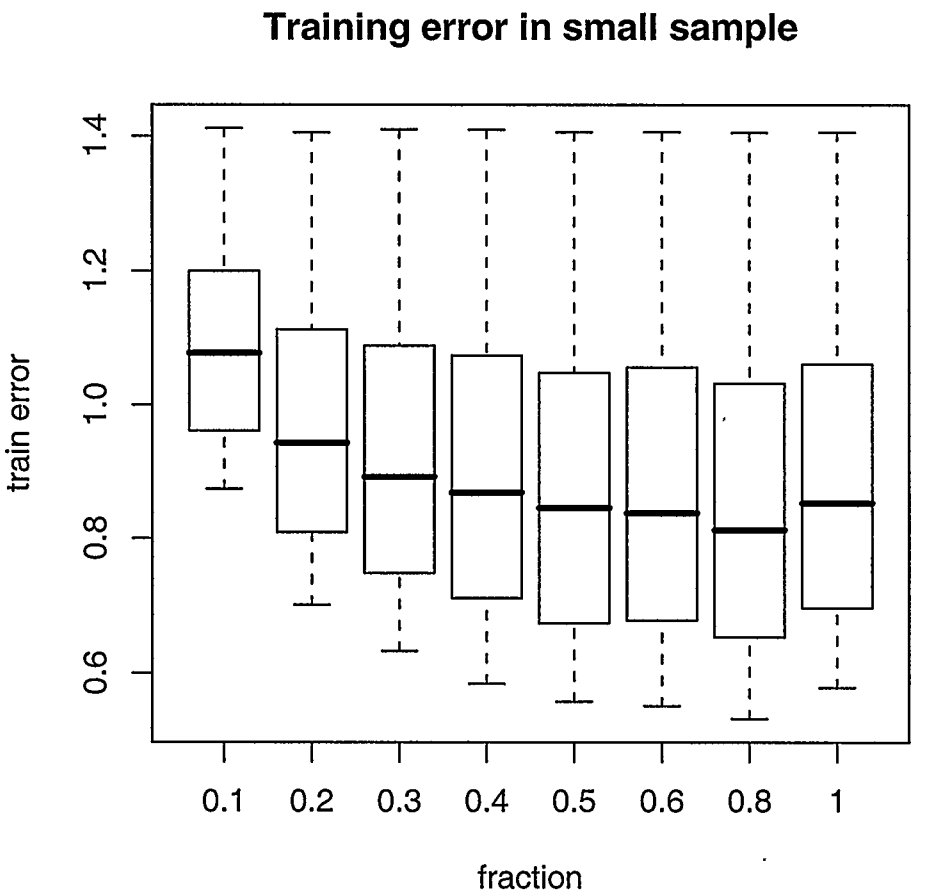

Figure 5.3: Training error under different random factors.

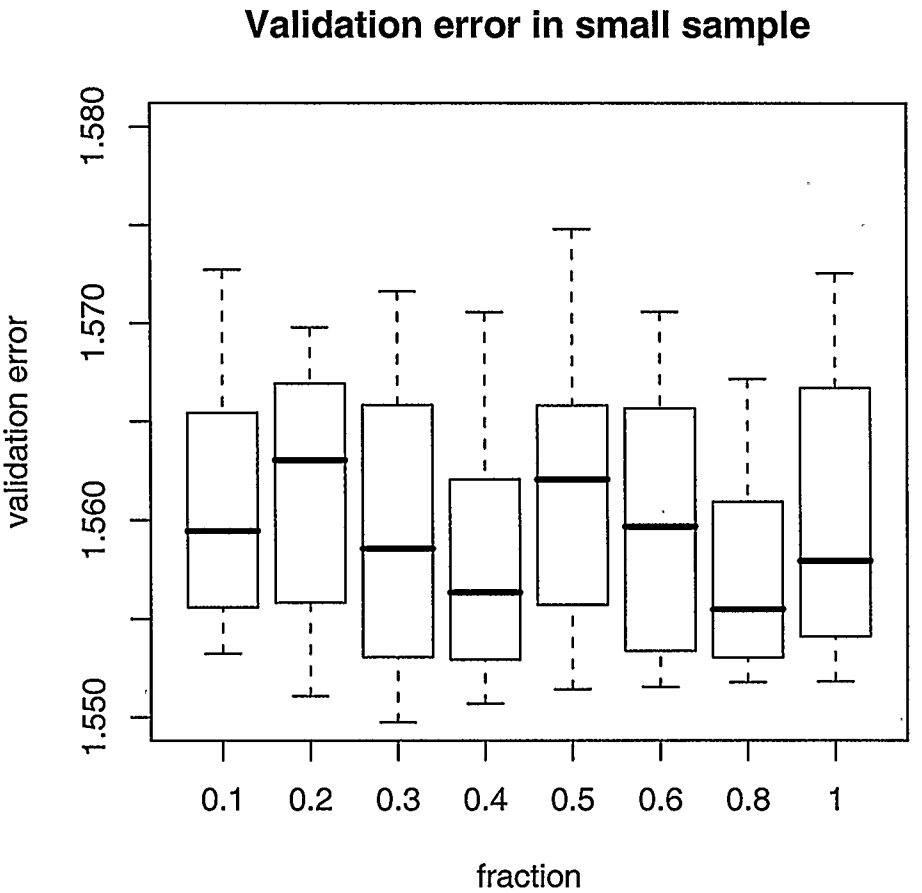

Figure 5.4: Validation error under different random factors.

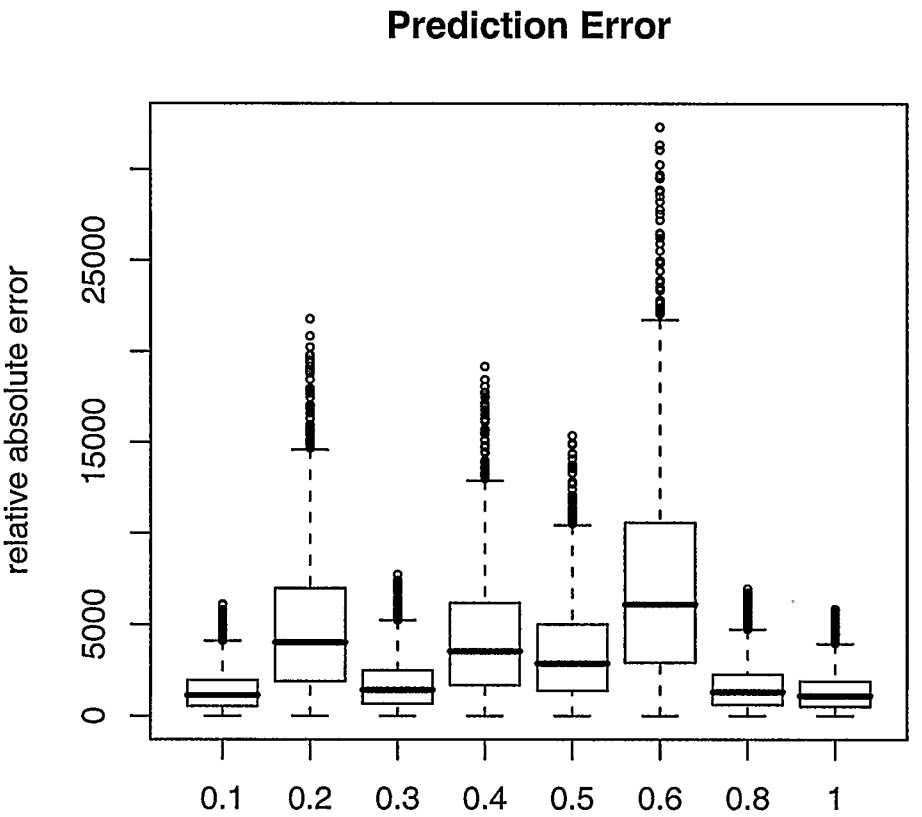

Figure 5.5: Relative error in prediction for large regression sample.

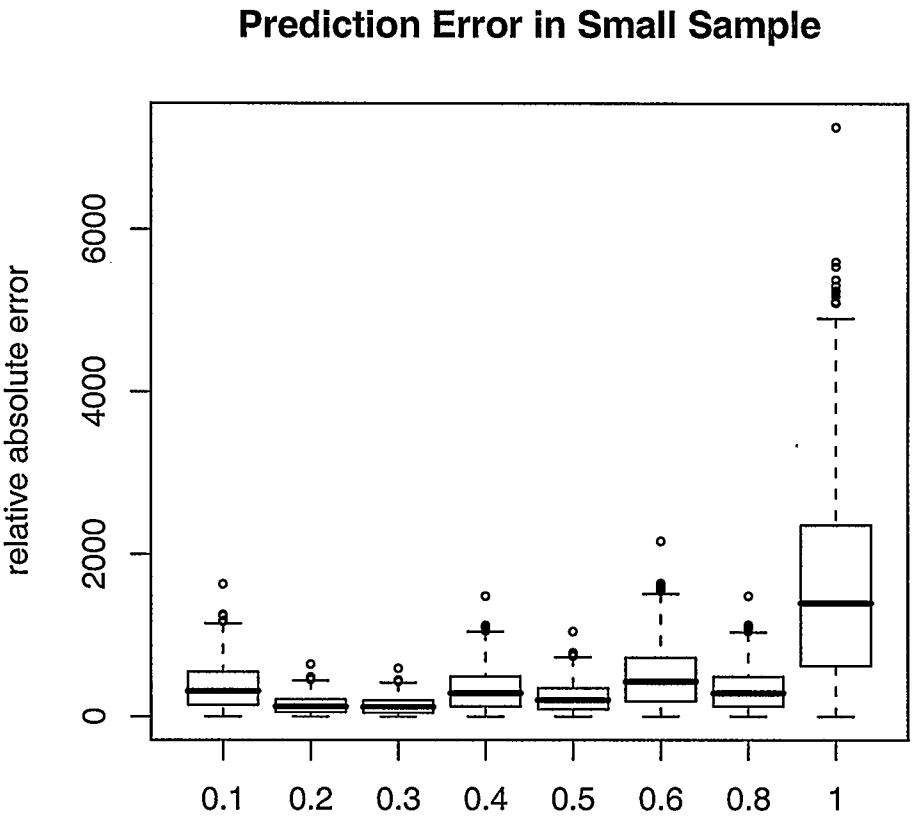

Figure 5.6: Relative error in prediction for the small sample case.

5.1.2 Classification

For the classification case, we use the Bernoulli distribution where the output of y belongs to $\{0, 1\}$. We convert the response values used in the regression section to 0/1 values by setting the new response equal to zero if $y <$ median (y) and to one otherwise.

Training error and validation error are computed using the default loss function in gbm, which is the deviance loss function (see Chapter 3). The plot of training error for the large-sample case (Figure 5.7) shows that the training error is roughly the same for values of q between 0.3 and 0.8.

The validation error plot (Figure 5.8) shows that $q = 0.4$ is a reasonable choice for subsampling fraction.

We use the misclassification rate as our statistical criterion for the test data. Figure 5.9 shows the misclassification rates of the models with different subsample fraction. Clearly, $q = 0.4$ is a good choice.

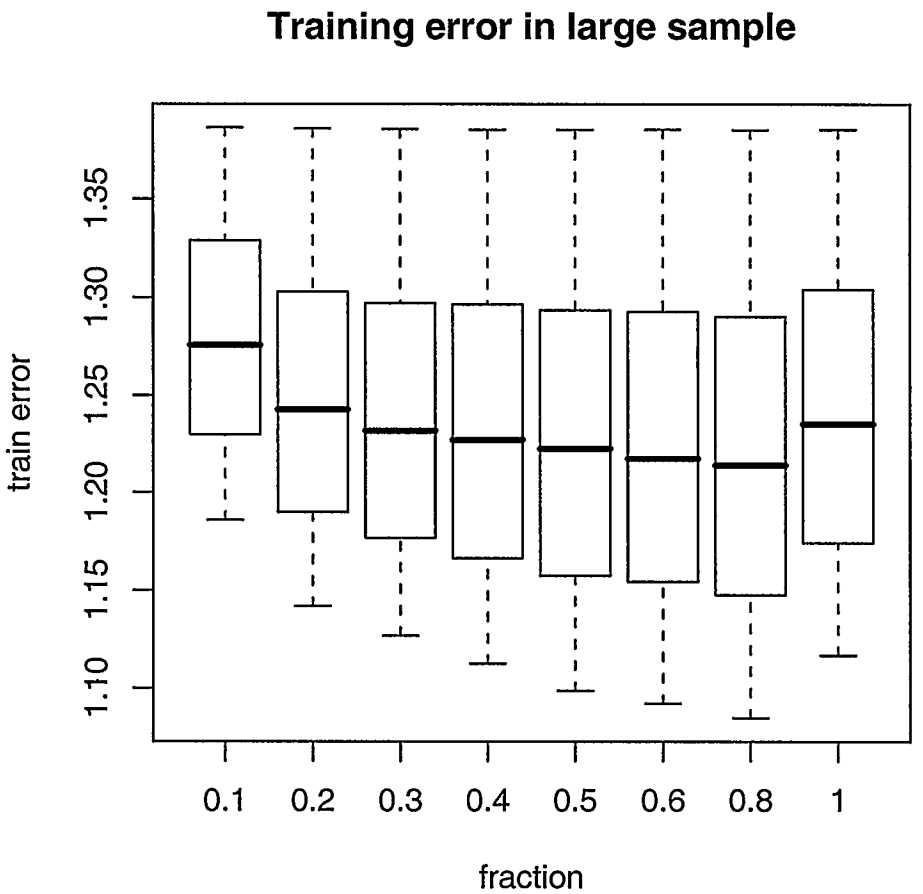

Figure 5.7: Training error under different random factors in large classification.

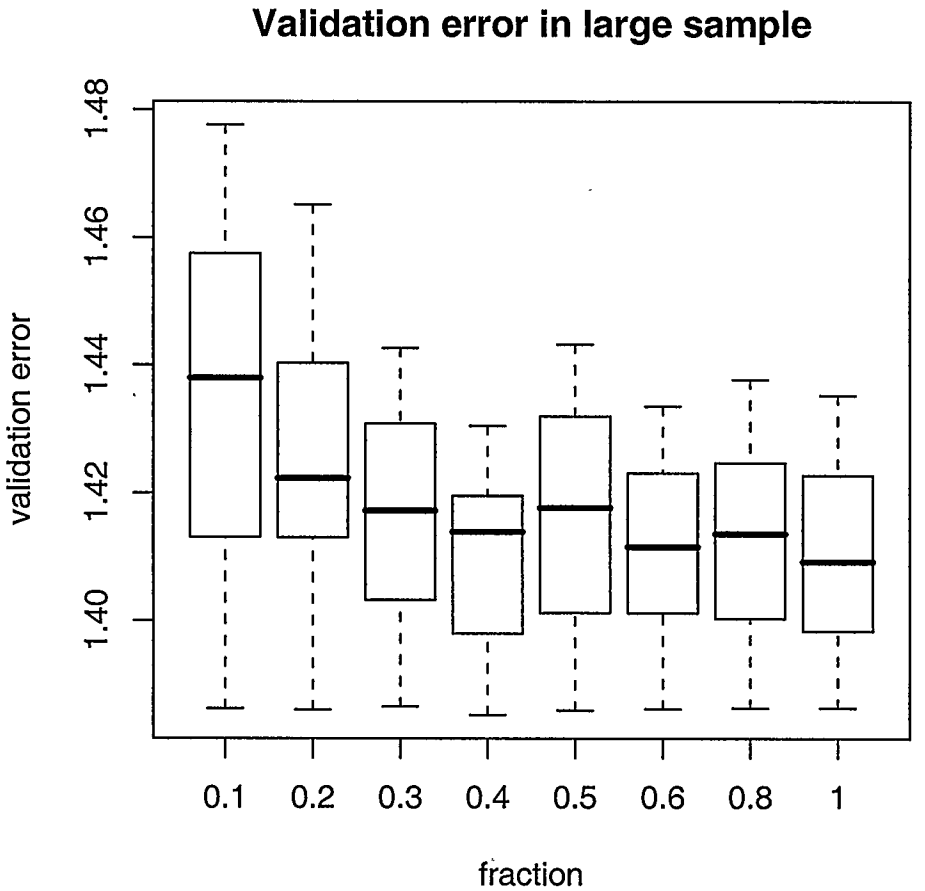

Figure 5.8: Validation error under different random factors in large classification.

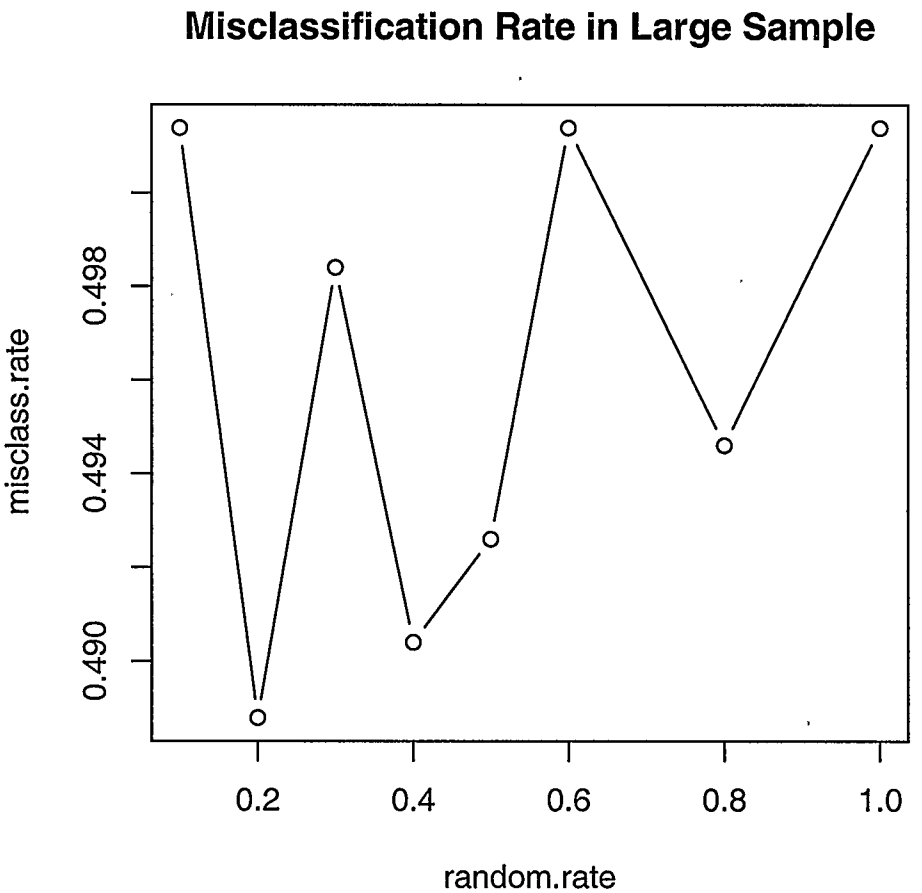

Figure 5.9: Misclassification rate using large sample.

For the small-sample data, Figure 5.10 shows that the training error is quite constant for $q > 0.3$.

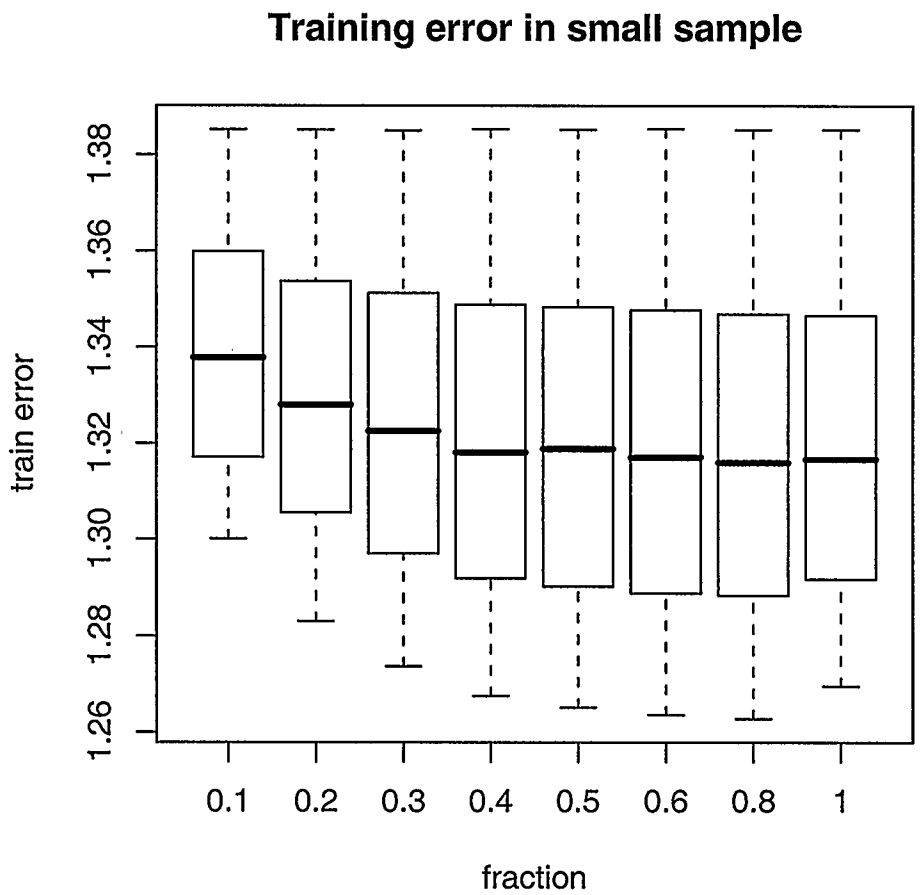

Figure 5.10: Training error under different random factors in small classification.

The validation error shown in Figure 5.11 indicates a U-curve trend with minimum response at $q = 0.5$.

The misclassification of 500 new data values is given in Figure 5.12. In this plot $q = 0.4$ generates the smallest misclassification rate. Combined with the previous

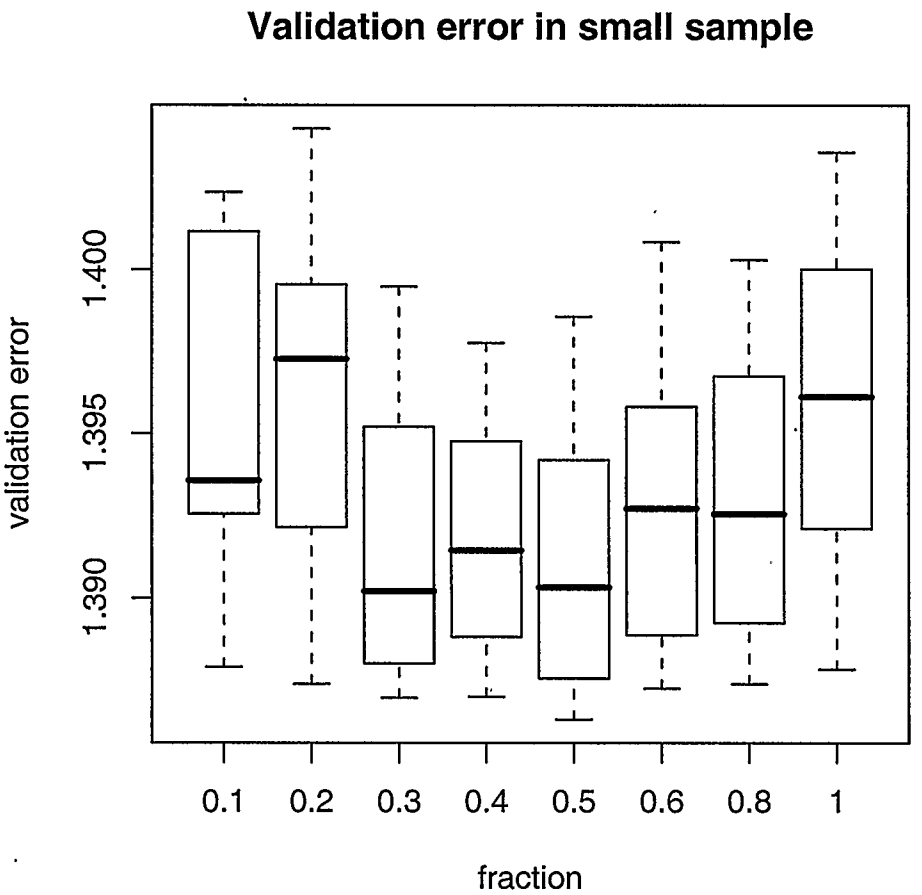

Figure 5.11: Validation error under different random factors in small classification.

two plots, $q = 0.4$ also has consistant performance in training and validation error. Therefore, we choose $q = 0.4$ as the optimal value in this case.

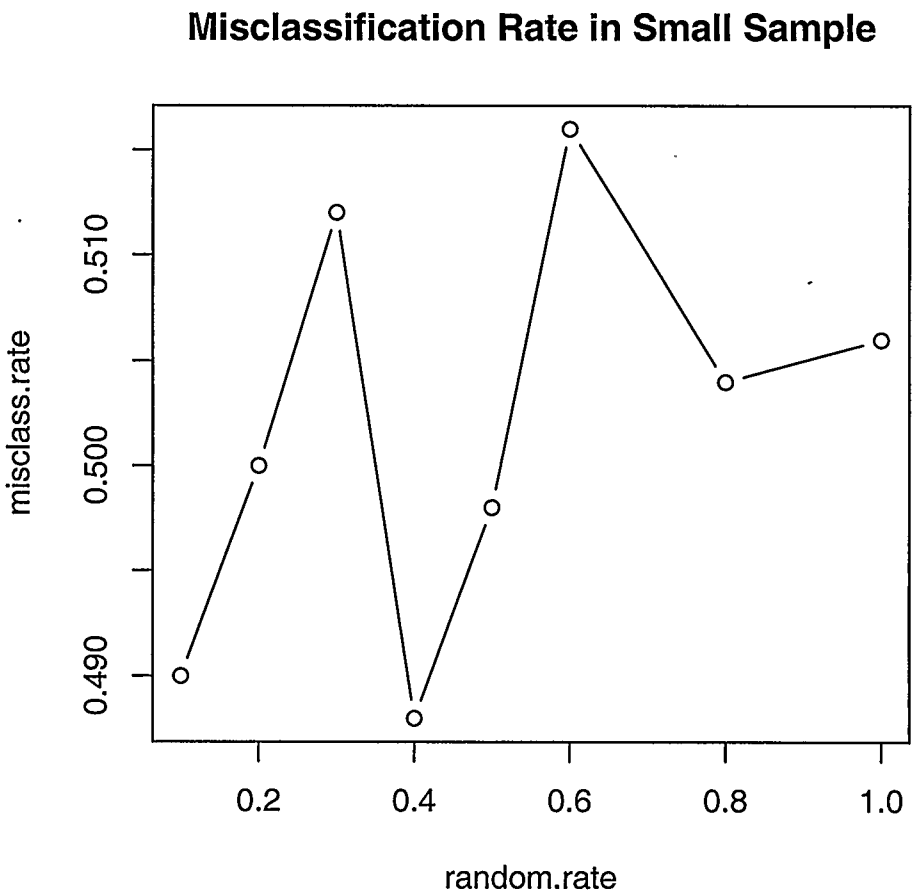

Figure 5.12: Misclassification rate using small sample.

5.1.3 Discussion

Randomness does work better than the deterministic algorithm but the degree of improvement depends on the case at hand [14].

Friedman [12] says,"the reason why this randomness procedure improves is not clear." Here, I attempted to give an answer as follows: in the context of bagging, randomly resampling with replacement absorbs the information of the full data as much as possible. But, because of replacement, the overlapping problem would make the model biased in prediction. Compared to bagging, stochastic boosting takes advantage of randomness but removes the replacement. Because each subsample is less than the full data, it is less likely to be as correlated with the other subsamples than would be the case if the full data were used at each iteration. Based on these subsamples, the model therefore should be better than the non-random one and should reduce overfitting.

The optimal value of the subsample fraction, based on the plots seen in the experiment, is $0.4 \leq q \leq 0.8$. However, if the sample size is too small, then q will have to be large in order that the model will be based on a reasonable number of observations. The degree of error reduction depends on the case at hand. Regardless of whether we are dealing with regression or classification, there are three factors that influence error reduction [12]:

1. the sample size N;

2. the distribution of the error term ε_i (we did not pursue this further);

3. the target function $F^*(x)$;

We should remember that the subsampling fraction should be adjusted together with the three regularization parameters (tree size, shrinkage, number of iterations). Balancing among these four parameters we can achieve an optimal model.

5.2 Interpretation

After the model is developed, the most important thing is to interpret it. Some applications, such as the single decision tree, are simple to interpret. The entire model can be a simple binary tree that can give us a visual and direct representation. This is why the decision tree is widely applied in finance and social economics. However the linear combination of trees

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m) \tag{5.5}$$

loses such an important property. Linear combinations are often not easy to be interpret, as, for example, in pricipal components analysis. Boosting trees suffer from the same difficulty.

We can try to use *relative importance* of predictor variables and *partial dependence* to explain the relationship between the inputs x and outputs y.

5.2.1 Relative Importance of Input Variables

A model of moderate size may have several variables, not all giving the same contribution to the target function. We would like to determine which variables have relatively more influence on the target function and then remove from the model those variables which are more or less irrelevant.

For a single decision tree, Friedman [12] defines a measure of relevance for each predictor variable X_ℓ as follows. For a tree T with J terminal nodes, let t index the nonterminal nodes. Let z_t^2 denote the model improvement when we create regions

R_l and R_r by the split at node t. We define the improvement as

$$z^2(t) = z^2(R_l, R_r) = \frac{w_l \times w_r}{w_l + w_r}(\bar{y}_l - \bar{y}_r)^2 \tag{5.6}$$

where $\bar{y}_l$, $\bar{y}_r$ are the mean response values for R_l and R_r and w_l and w_r are weights for the two regions.

To assess the relative influence of a predictor, we find all internal nodes split by that predictor and sum the improvements at those nodes. Let u_t index the splitting variable at node t. Then the squared relative influence of X_ℓ is

$$I_\ell^2(T) = \sum_{t=1}^{J-1} z_t^2 I(u_t = \ell) \tag{5.7}$$

The above definition applies for a single tree. In boosting, the iterations generate many trees and a natural generalization of the single-tree squared influence is its average over all the trees:

$$I_\ell^2 = \frac{1}{M}\sum_{m=1}^{M} I_\ell^2(T_m) \tag{5.8}$$

Applying the above definition to the small simulated dataset ($N = 1500$) and using the least-squares loss function in `gbm`, we obtain Figure 5.13. (The plot is obtained from the "`summary()`" command.)

The left plot in Figure 5.13 is for a single iteration. It shows that x_6, x_4, x_{10}, x_3 and x_5 are dominant among all 10 variables while the other 5 variables have negligible influence. However, after completing the optimal number of iterations (47), more variables contribute to the target function, creating a more complicated model. Nevertheless, the target function is still dominated by a small subset of all the variables. As can be seen in the plot on the right of Figure 5.13, x_3, x_4 and x_{10} still contribute considerably to the model, albeit with changed rank order. But the

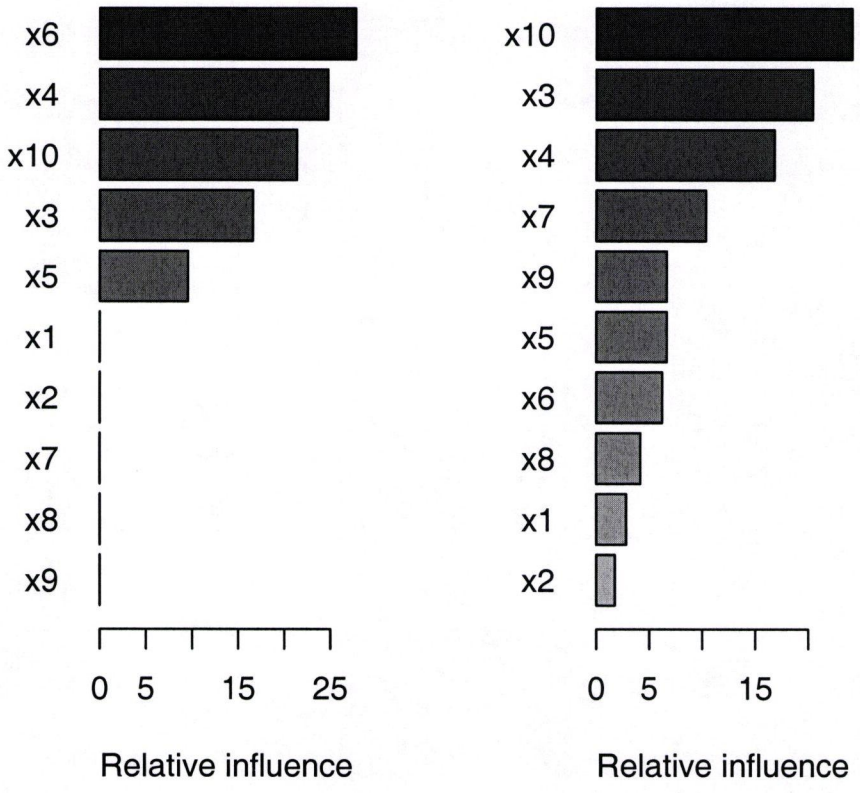

Figure 5.13: Contribution from all the variables to the target function based on one iteration (left) and the best number of iterations (right).

influence of x_5 is somewhat reduced and that of x_6 is much less. Of the remaining variables, x_7 has achieved moderate relative influence while the others have seen smaller increases. The model is now mainly a function of x_{10}, x_3, x_4 and x_7, in the order of importance. That a fairly small number of predictors dominate the model is consistent with the claim by Friedman [12, p. 1208] that "target functions often strongly depend only on a small subset of all of the inputs."

5.2.2 Partial Dependence

After finding the influential variables, we need to understand the dependence of the target function f on these variables. Graphic tools are useful in this endeavour. For a single real-valued input x, the response values $f(x)$ can be plotted against x; for a single categorical input variable, box-plots may be used. Contour plots are useful to illuminate the dependence of f on two real-valued variables at a time. Trellis plots may be used to summarize the dependence of f on a pair of input variables, one real-valued and the other categorical.

These graphic representations are limited to low-dimensional functions. Fortunately, as we have seen above, although there might be many input variables, the number of highly relevant ones is often limited to just a few.

A partial dependence plot shows how the target function depends on a subset of the predictors (usually one or two).

Following Hastie *et al.* [16], we consider the input vector $X = (x_1, x_2, \ldots, x_p)$ and generate subvectors X_a and X_b, where $X_a \bigcup X_b = X$. Then the target function can

be witten as $f(x) = f(x_a, x_b)$. The partial dependence of $f(x)$ on x_a is

$$f_a(x_a) = E_{x_b}[f(x_a, x_b)] = \int f(x_a, x_b) p_b(x_b) dx_b \tag{5.9}$$

This is the marginal average of $f(x)$. It is important to realize that the partial dependence functions represent the effect of x_a on $f(x)$ after accounting for the (average) effect of x_b on $f(x)$. Equation 5.9 can be estimated from the training data as

$$\bar{f}_a(x_a) = \frac{1}{N} \sum_{i=1}^{N} f(x_a, x_{ib}) \tag{5.10}$$

where $x_{1b}, x_{2b}, \ldots, x_{nb}$ are the values of x_b occurring in the training data.

In Figure 5.14, we illustrate the use of partial dependence plots with the variables shown in the left plot of Figure 5.13. Figure 5.14 shows the single-predictor partial dependence plots for the four influential predictors.

None of the four plots is monotone, but we see that the effect of x_3 is essentially decreasing while, for x_4 and x_{10}, there is roughly an increasing effect. The plot for x_6 approximates a step function.

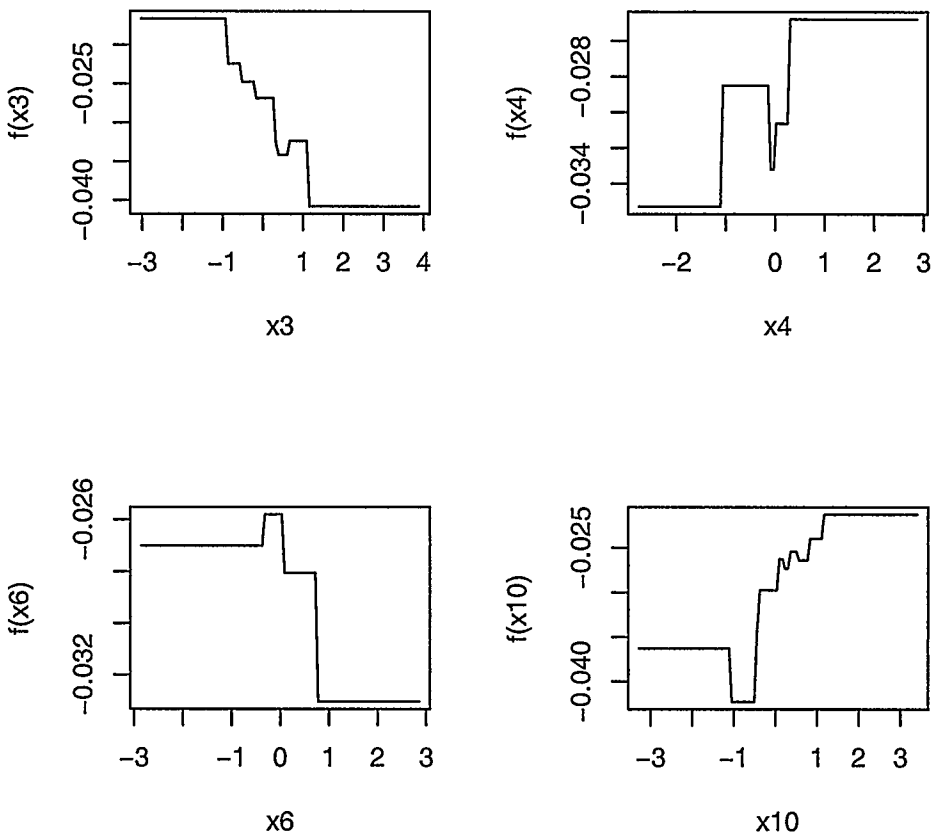

Figure 5.14: Single-predictor partial dependence plots of selected variables.

Figure 5.15 shows how the target function depends on x_4 and x_{10}, simultaneously.

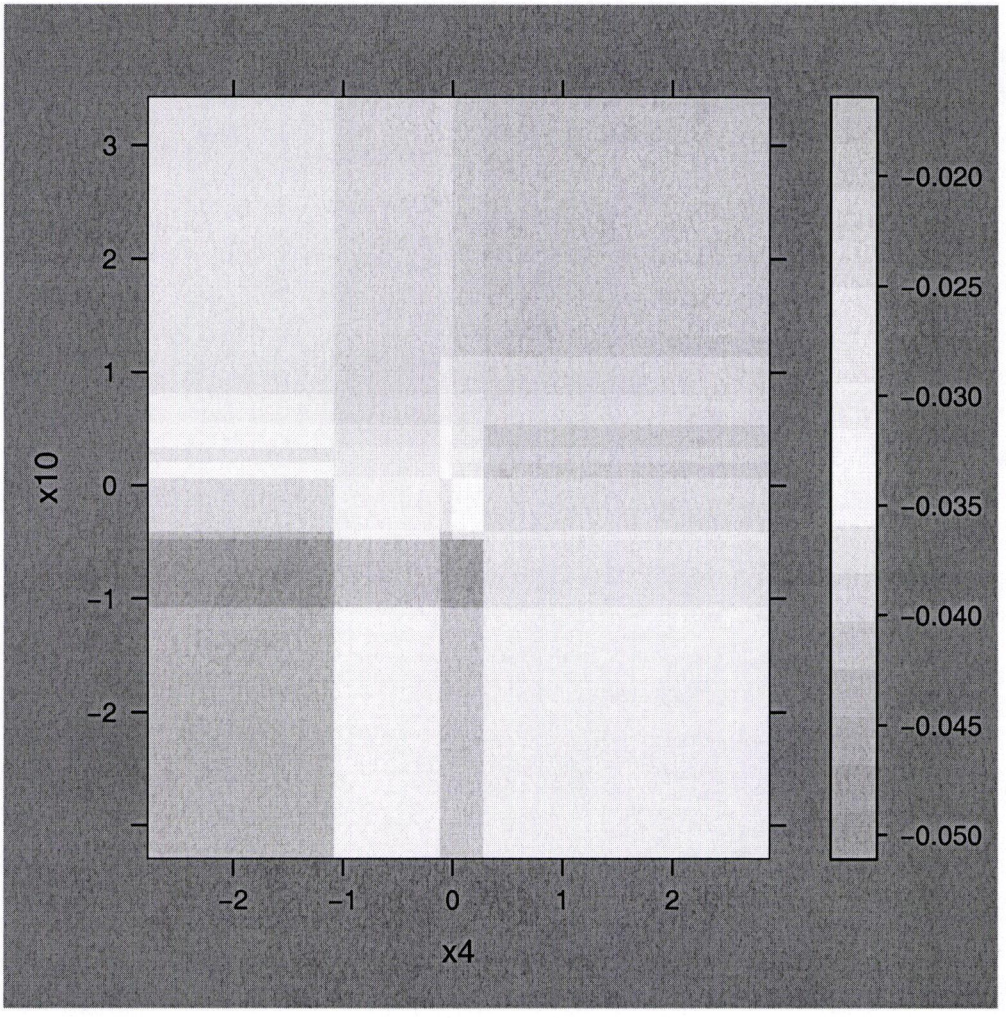

Figure 5.15: Partial dependence of the target function on two predictors.

Chapter 6

Applications of the Boosting Algorithm to Real Data Analysis

Researchers in different fields have realized the importance of statistical tools, and some of them have started applying the boosting method to biology, medical science and engineering for both classifications and regressions. To our best knowledge, however, the boosting algorithm has not been used for studies in social science and management science yet. To convince the readers, we apply it to a financial dataset, entitled "Financial Conditions in Small Canadian Businesses", collected by Industry Canada in Year 2001, and study some debt financing issues of Canadian small and medium enterprises (SMEs). We also compare the boosting algorithm with the bagging method and illustrate the advantages of the former in both training error reduction and computational efficiency.

Besides being a pioneer work in applied statistics for the boosting method, two major reasons induce us to conduct the real data study using this dataset. First, debt financing is one of the two major ways for a firm to raise money, while the other is equity financing. For small businesses to finance growth, debt financing is dominantly important since they have very limited accessibilities to the public market and most of them are not qualified to issue public equities. Therefore, studying debt financing issues for Canadian SMEs is very meaningful to the Canadian economy since over 97.7 percent of the 2.2 million firms in this country by 2001 have been categorized

as small and medium-sized businesses. These SMEs employed more than 10 million individuals and contributed 43% of private sector Gross Domestic Product (GDP) in 2000[1].

Second, this data set helps us to highlight the advantages of the boosting algorithm over the relatively traditional bagging method and those over the logistic model or the OLS regression, in both prediction accuracy and computational efficiency. Furthermore, this data set is appropriate for studying debt financing issues for Canadian SMEs since it is nationally representative and industry-wide. It also covers different aspects of firm situations, including financial variables, agency variables, firm characteristics, firm owner's characteristics, and lender's characteristics. These variables have been widely used in the finance literature. Applying the boosting algorithm to this data set can also explore two important issues in Canadian SME debt financing without potentially losing consistency of statistical tools. The first is whether a small firm applied for debt financing, and the second is how much loan they obtained if the application was approved. Whereas the former is studied using boosting classification because of the binary measure of the dependent variable, the latter is done with boosting regression. The focus is to explore the influences of agency factors on small business debt financing.

6.1 Data and Variables

This dataset consists of part of the variables from a survey, entitled "Financial Needs of Canadian SMEs", commissioned by Industry Canada in Year 2001, and the pur-

[1]Data Source: *Small Business Research and Policy.* Published by Industry Canada in July 2002. http://strategis.ic.gc.ca/epic/internet/insbrp-rppe.nsf/vwGeneratedInterE/rd00668e.html

pose of this survey is to better understand the financial conditions of Canadian SMEs. The overview period was for 1998–2000, with a focus on the first six months in 2001 and financing intentions for the upcoming year.

From a total of 10,020 mail and telephone surveys 2,148 valid responses were received. For various reasons, 32 surveys were dropped leaving a total of 2,116 observations. The survey covered various vehicles of financing but, following mainstream literature, we concentrate on existing bank lines of credit (LOCs). As indicated by Berger and Udell [1], line of credit is the most popular way for small firms to obtain short-term debt financing, and they usually obtain long-term debt financing through renewing short-term financing. Industry Canada had suggested representation of firms by sector of activity and size. Thus the statistics presented in the study were based on adjusted weightings reflecting such factors. The weighted sample has 2,027 observations. One of the most serious problems existing in the data set is the missing value problem, which may generate bias in the analysis. To fit the requirement of boosting, we dropped those observations with a large number of missing values, resulting in a final data set of 1,896 observations.

In the corporate governance literature, agency theory is one of the fundamental issues, and its influences on debt financing of large, publicly listed firms have been extensively studied. In the finance and small business lending literature, however, this issue is still at its early stage. While we are illustrating the advantages of the boosting algorithm in its training error reduction and computational efficiency, we also try to add to the finance and small business literature regarding how agency variables, firm characteristics and borrower's characteristics affect small business debt financing. To further advance the literature, we study two dimensions of debt

financing, the accessibility of small firms to debt financing and the amount approved by lenders. The former is measured by whether a small firm applied for LOCs during the period 1998– 2000 (APPLY), while the latter is measured by the debt-asset ratio in Year 2000. The variable APPLY is binary; a value of 1 was assigned to firms that had applied for LOCs during the period 1998–2000, and zero otherwise. As a continuous variable we use the debt ratio (LEVERAGE). A descriptive analysis indicates that more than 45% of the Canadian SMEs did not apply for LOCs during that period, and the average leverage was 48.43% in Year 2000.

Following the finance and small business literature, we include 12 indenpendent variables that can be categorized into three groups, the agency variables, the firm characteristics, and firm owners' characteristics.

Agency Variables In corporate governance, monitoring, screening and signalling are three most popular solutions to informational asymmetry problems, which can generate agency issues. As indicated by Berger and Udell [1], relationship-lending is critical in small business debt financing, and this is considered one of the most significant differences between SMEs and large, publicly listed companies. In the data set under study, only information about screening is not available. Due to the time sequence problems in relationship variables[2], however, we include only signaling and monitoring variables in the current study.

Singaling is one of the most useful methods dealing with asymmetric information problems. Since the informational asymmetry is much more serious between small

[2]Since the survey was done in Year 2001, firm owners might also have taken the results of their applications and the amount of approval into consideration when they answered questions about their relationship with institutional lenders. In other words, relationship variables can reflect the accessibility to LOCs and the amount of approval.

firms and banks than those between large, pulicly listed firms and banks, signaling becomes more important in small business lending. We follow the literature and use two variables to measure signaling issues. The first variable, PERSONAL, is measured by whether the owner used his personal credit record to help finance the business in the past three years. Whether the owner would like to put his personal wealth at risk along with the business is a strong signal of the owner's confidence to the business. The second variable, REPORT, is a dummy variable indicating whether there is a person, other than the owner, taking charge of the finance and accounting affairs. This person can be regarded as the "third person" who takes care of the finance and accounting in the business in a timely and accurate fashion. Traditional corporate finance papers suggest that having a person in charge of finance and accounting matters signals the importance the firm attaches to such matters.

The frequency of board activities is a variable measuring monitoring issues. From the lender's point of view, moral hazard problems are hard to be monitored by outsiders so that the board of directors take the role from inside. Frequent board meetings will be good for lenders to control agency problems, although inefficiency and cost-consuming problems may affect firm performance. In small businesses, specifically, the frequency of board meetings should not be very high since they cannot afford the high costs, unless the lagged firm performance was poor or some unexpected affairs occur. BOARD, the indicator variable measuring board meeting frequency, takes the value one if a firm had board meetings more than twice a year during the period of 1998–2000; it has a value of zero otherwise.

Firm Characteristics: The variable SOLE is used to measure the firm's legal organizational form and to tell whether the borrower is the only owner of the firm

or not. Since there is more legal separation between personal and firm assets in the corporation, the legal form of the business may be important. An owner who created a business will be more committed to the survival of the firm, especially when he is the only creator. Furthermore, if he also gets involved in the management, his behavior toward the risk would be much different; hence we introduce the indicator variable MANAGE.

We also introduce "the number of employees" (EMPLOYEE) to measure firm size and "the age of the firm" (AGEFIRM) to measure the firm's stage of development. We include the "age of firm" because it would tend to give positive effect to the financing. It is obvious that an older firm with longer credit history would be more likely to access debt financing than a younger business with shorter credit record.

Borrower's characteristics: In addition to the variables defined above, the borrower's personal information will contribute to the risk exposure of the lender, so it may affect both the credit evaluation and the intention of borrower to continue applying debt. Previous research has shown that female entrepreneurs generally find it difficult to obtain capital. Their results generally find that lending discrimination is not related to these factors directly but rather appears to be driven by a lender's reluctance to extend credit to the types of firms often established by these groups. These firms may be too small, have less profit, have a poor credit record or be unwilling to provide personal guarantees with collateral. Hence we have the variable GENDER (male or female).

Language is also included here because there is still lingering suspicion in Canada that the main language used in the enterprise may be the basis of discrimination in lending. We introduce the indicator variable ENGLISH as our language effect.

In addition to all the variables mentioned above, we also adopt "years of experience" as a predictor of success. We split this risk measure into "firm experience" (EXPFIRM) and "industry experience" (EXPIND). Age of the owner (AGEOWNER) is also concerned.

The variables are summarized in Table 6.1.

Variable	Type	Definition
PERSONAL	Indicator	used personal credit record?
REPORT	Indicator	other person in charge of finance?
MANAGE	Indicator	owner involved in management?
AGEFIRM	Numerical	age of firm
SOLE	Indicator	single owner
EMPLOYEE	Numerical	number of employees
GENDER	Indicator	male/female
ENGLISH	Indicator	primary language: English/French
EXPFIRM	Numerical	years of experience with firm
EXPIND	Numerical	years of experience in industry
BOARD	Indicator	more than two meetings?
AGEOWNER	Numerical	age of owner

Table 6.1: Summary of variables.

For the numerical response of debt-asset ratio (LEVERAGE), the data set yields 452 complete cases to which we can fit a boosted regression tree model. We used 160 cases for the training data and 40 cases for validating the model. The remaining 252 cases were used for testing the model.

For the indicator response APPLY, we have 1896 complete cases which we use for a boosted classification tree model. We used 500 cases as the training data, and the rest as the testing data.

In the following section, we will conduct robustness tests showing the effectiveness

of the boosting algorithm in training error reduction.

6.2 Robustness Experiments

6.2.1 The Boosting Regression Case

Since LEVERAGE is a continuous variable, we set up a boosting regression model under squared-error loss, with a shrinkage 0.05 and stochastic parameter 0.1. This means that a random subset of 10% of the training sample was used at each iteration to build the model. The other 90% constitute the "out-of-bag" data.

Figure 6.1 shows the resulting training error and validation error curves. We find a trade-off between training error and validation error. Clearly, 200 iterations are far too many, since the validation error is then large. The small training error and large validation error for 200 iterations signals an overfitting problem.

In order to find the optimal number of iterations to balance the training error and validation error, we followed Ridgeway's [22] suggestion and first obtained an out-of-bag estimate (M_{OOB}) for the optimal number of iterations. Then, if the difference between the current number of iterations in the model and M_{OOB} was less that 10, we added 100 more iterations. The final optimal number of iterations was then obtained using the validation data.

We find an optimal number of iterations is 27. Also shown in Figure 6.1 is the out-of-bag improvement in squared-error loss as a function of the number of iterations. Note that this improvement is maximized at about 27 iterations.

After finding the optimal number of iterations, we measure the effects of independent variables on the leverage taken by Canadian SMEs. Among the 12 independent

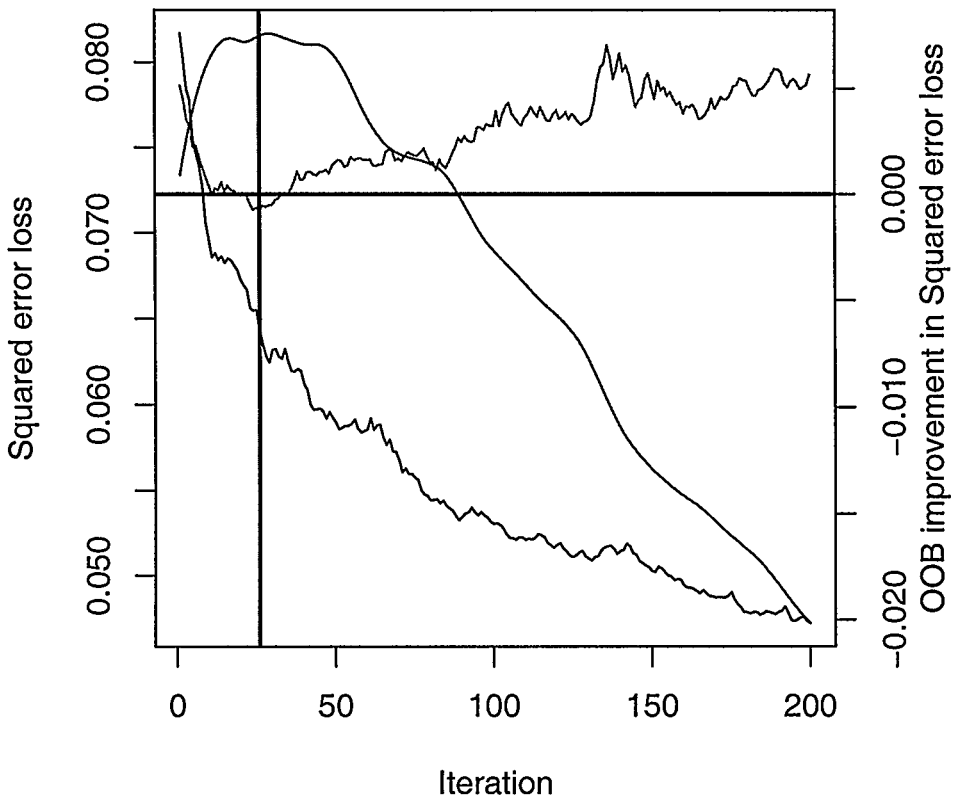

Figure 6.1: Optimal number of iterations.

Order	Variables	Relative influence
1	AGEOWNER	28.97
2	EXPFIRM	26.72
3	EXPIND	20.91
4	AGEFIRM	17.65
5	EMPLOYEE	12.75
6	PERSONAL	< 0.0001
7	GENDER	< 0.0001
8	BOARD	< 0.0001
9	ENGLISH	< 0.0001
10	REPORT	< 0.0001
11	SOLE	< 0.0001
12	MANAGE	< 0.0001

Table 6.2: Relative influence for regression.

variables, 5 have dominating effects according to their relative influences; they are: age of owner, firm experience, industry experience, age of firm, and number of employees. In short, age, experience, and firm size are the most important factors determining small business debt financing. These results are illustrated in Table 6.2 and in Figure 6.2.

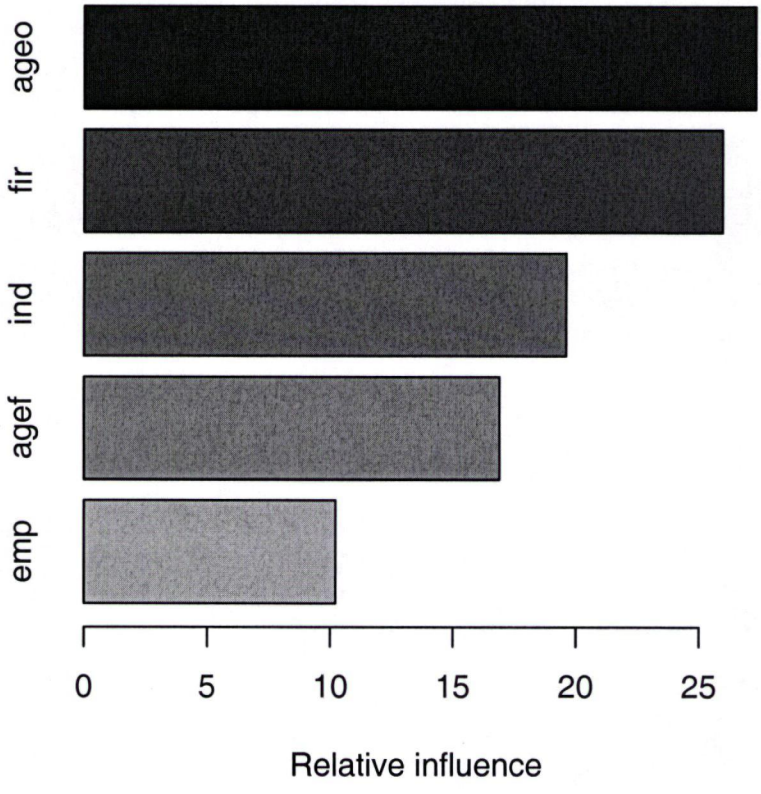

Figure 6.2: Relative influence of predictors.

Table 6.3 illustrates the comparison betwen the results from the boosting regression and those from the bagging method. If we use 300 iterations, disregarding the overfitting problem, the root mean square error for boosting is higher than that for bagging. Bagging performs better by 10.67%. If we use 27 iterations, the optimal number for boosting, then the rms error for boosting is slightly smaller than that for bagging; boosting performs better than bagging by 1.9%. However, boosting performs much better than bagging in reducing training errors. The mean error, maximum and minimum errors from boosting are all significantly lower than those from bagging.

Criterion	Bagging		Boosting	
Number of iterations	300	27	300	27
Root mean squared error	0.275	0.275	0.308	0.270
Training error reduction	based on 300 iterations			
Mean	0.294		0.056	
Maximum	0.306		0.082	
Minimum	0.283		0.046	

Table 6.3: Boosting compared with bagging.

6.2.2 The Boosting Classification Case

1,896 observations were included in the boosting classification for the applications for LOCs, and we operated these data with 500 iterations under binomial deviance loss function. Shrinkage was set at 0.05 and the stochastic sampling proportion was set at 0.1.

After computing the relative influences of the 12 independent variables, only the gender of the firm owner has entirely negligible effect on the binary dependent

Order	Variables	Relative influence
1	Number of employee	21.66
2	Age of owner	19.26
3	Industry experience	15.43
4	Age of firm	13.97
5	Firm experience	11.65
6	English	5.35
7	Report	4.15
8	Personal	3.15
9	Board	2.29
10	Sole	2.00
11	Manage	1.09
12	Gender	< 0.0001

Table 6.4: Relative influence for classification.

variable APPLY, while the five variables dominating others in the boosting regression model still play important roles in the boosting classification model. These results are tabulated and illustrated in Table 6.4 and Figure 6.3.

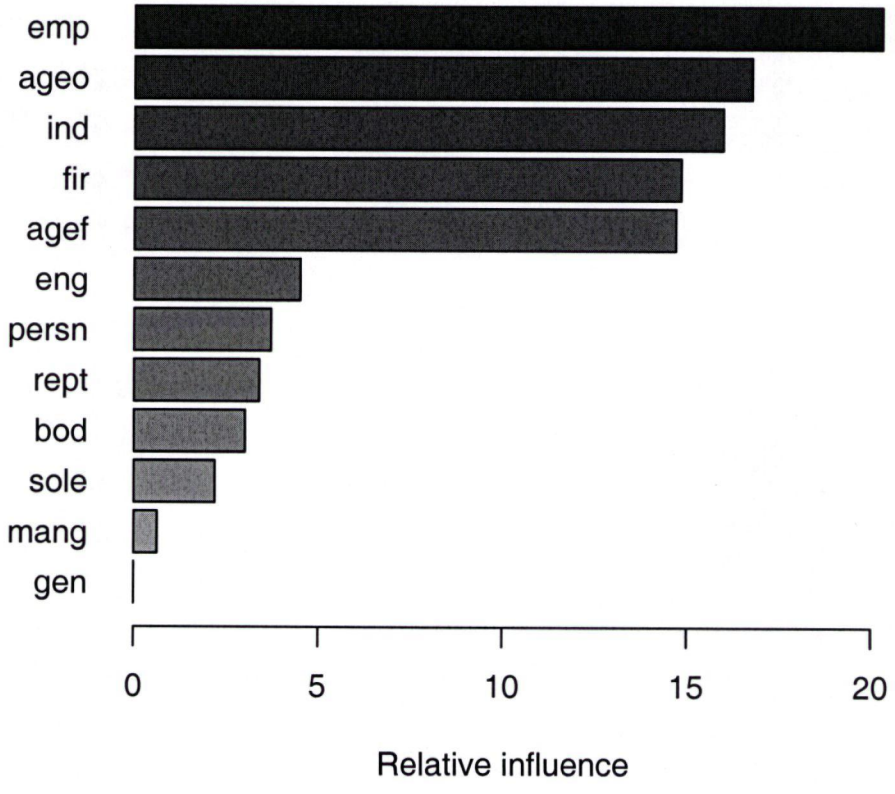

Figure 6.3: Relative influence of predictors for classification.

The optimal number of iterations in the boosting classfication to balance the trade-off between training error and validation error is about 82, which was found by the same method as for the boosting regression case.

Besides the training errror reduction, we can explore the relationship between important predictors and our target functions. Figure 6.4 reveals this relationship for the most influential five predictors. Among those five plots, we can see that "age of owner" (**ageo**), "industry experience" (**ind**) and "firm experience" (**fir**) have roughly similar plots where firms with low values of the predictor are more willing to apply for debt. The plots for "number of employees" (**emp**) and for "age of firm" (**agef**) are also qualitatively similar, showing that large or older firms are very willing to apply for debt.

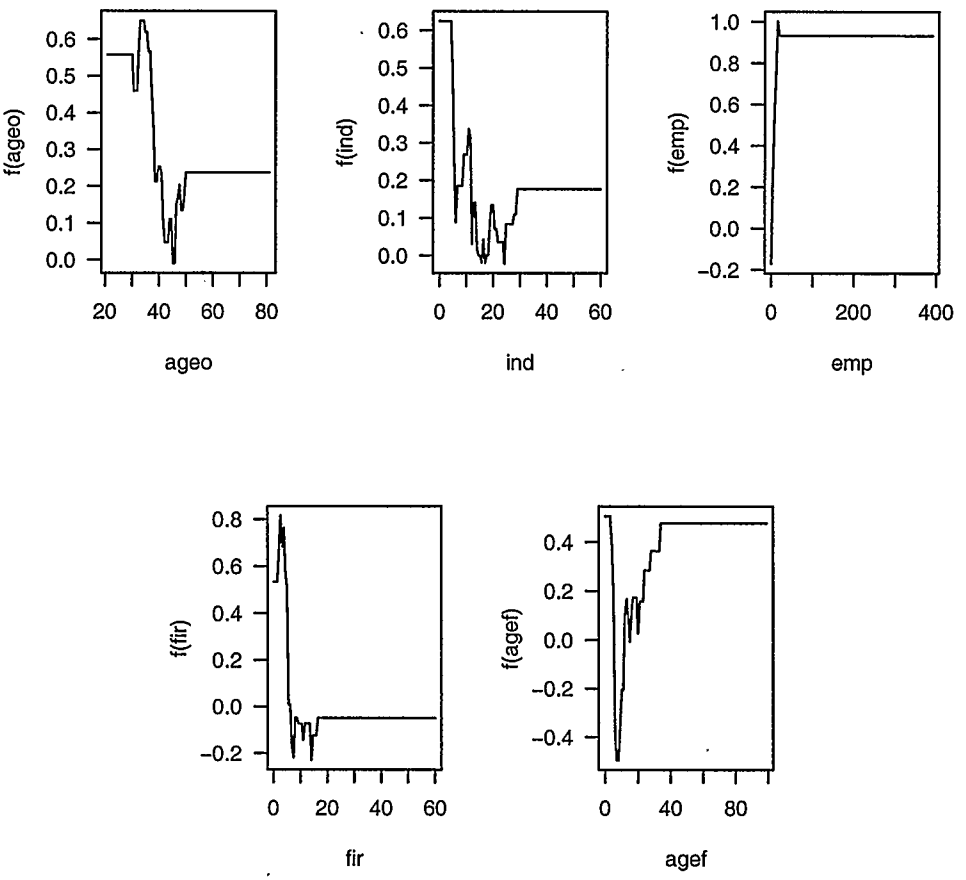

Figure 6.4: Partial dependence of selected predictors.

To illustrate the prediction accuracy of the boosting algorithm, we also compare the results from the boosting classfication model with those from the bagging method. The average misclassification error of bagging under 500 iterations is 0.3855 with a 0.0024 standard deviation, while the misclassification error of boosting under 500 iterations is 0.3788. For 182 iterations, the average misclassification error is 0.3656. This illustrates the existence of overfitting problems. When we run too many iterations, such as 500, the misclassification rate increases, but it still improved the prediction from bagging by 5.17%.

Boosting works better than bagging in both classification and regression cases, but unfortunately, the improvement is not as significant as what we have seen using simulation data. However, taking the computational efficiency into consideration, boosting is absolutely better than bagging.

6.3 Summary

We have seen that boosting performs quite well and better than the bagging method. It reduces error better in prediction than bagging does, although under some circumstances, the improvement is slight. The boosting algorithm can balance the trade-off between training error reduction and validation error well, and its advantages in model prediction include both estimation accuracy and computational efficiency.

Applying the boosting algorithm to small business debt financing, we find that firm size is one of the most important variables determining the accessibility of small firms to short-term debt financing and the leverage taken by them. Other key variables include age of the firm, age of the owner, firm experience and industry

experience of the firm owner.

Chapter 7

Conclusion and Further Study

Boosting represents a new class of learning methods, introduced in the last ten years. The structure of boosting can be regared as iterated additive model. As the number of iterations increases, more weak learners are added into the target function. Because of weights at each iteration, those weak learners keep being improved until the whole procedure stops. After averaging weak learners, boosting can produce a strong learner.

In previous chapters, we displayed the effect of boosting in model fitting and prediction. However, the advantage of doing boosting also depends on the case at hand. If the weak learner employed in boosting is just a stump (one-split tree), for example, then boosting can give significant improvement. However, if the "weak" learner is a larger tree with 8 nodes, then boosting cannot change the model too much because an 8-node tree is a not weak model in most cases.

The choice of values for the parameters also affects the performance of boosting. Based on the size of the dataset, the number of iterations is an easy parameter to optimize. If the data set is learger than 10000, then approximately 500 iterations whould be a reasonable choice. Other parameters, such as the tree size, subsample fraction and shrinkage, also help to balance model fitting and prediction.

Compared to similar algorithms, such as decision trees and bagging, boosting can give considerable appreciation in accuracy, producing a computational saving as well.

But boosting is still a quite new learning algorithm. Thus, there is still capacity for research into improvements to the algorithm. Quite a few research papers on boosting have been published recently.

I did not find boosting to be as advantageous as I had hoped. The results reported by Friedman [13] are much more impressive than those I found for the Canadian Small Business data. I found that the improvement over bagging was not large. But the performance of any algorithm depends on the data at hand, and the data I used was perhaps not sufficiently well structured to permit a good model. Nicholas [20], Kutin and Niyogi [19] and Gey and Poggi [15] all mention a problem with stability of boosting models.

In my real data study, I omitted cases with missing values. This could easily be avoided, and the results perhaps improved, by using some data imputation method.

Boosting is becoming a more mature algorithm, both from a theoretical view and in application. I only applied binary classification in my study, but boosting also can deal with multi-class cases. Some work has been done by Friedman *et al.* [14]. It still starts off by a natural generalization of the two-class symmetric logistic transformation, and then the multiclass boosting algorithm maximizes the corresponding Bernoulli likelihood for each class versus the others. The multiclass algorithm has been tested on simulation data and also gave significant performance. To date, `gbm` cannot deal with multiclass problems. This would be a useful addition to work on.

Moreover, some researchers have applied boosting to game theory and survival model analysis and more related research has been reported by Ridgeway [22], Nicholas [20], Schapire [27]. These two topics introduce boosting into a new area to deal with more realistic problems. Game theory is an application of classification,

while proportional hazards regression in survival models also can be regarded as regression case in boosting. In addition, we expect that boosting can find significant applications in economics and biology/medicine.

Bibliography

[1] Berger A. and Udell G. (1995). Relationship lending and lines of credit in small firm finance. *Journal of Business*, **68**(3): 351–382.

[2] Breiman L. (1994). Bagging predictors. Technical Report 421, Department of Statistics, University of California, Berkeley.

[3] Breiman L. (1996). Bagging predictors. *Machine Learning*, **24**(2): 123–140.

[4] Breiman L. (1997). Prediction games and arcing algorithms. Technical Report 504, Department of Statistics, University of California, Berkeley.

[5] Breiman L. (1998). Arcing classifiers. *Annals of Statistics*, **26**: 801–849.

[6] Buja A. and Lee Y. (2001). Data mining criteria for tree-based regression and classification. In *Proceedings of KDD*, pages 27–36.

[7] Esposito R. and Saitta L. (2003). Monte Carlo theory as an explanation of bagging and boosting. In G. Gottlob and T. Walsh, eds., *Proceeding of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 499–504. Morgan Kaufman.

[8] Freund Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation*, **121**(2): 256–285.

[9] Freund Y., Mansour Y. and Schapire R.E. (2001). Why averaging classifiers can protect against overfitting. In *Proceedings of the Eigth International Workshop on Artificial Intelligence and Statistics.*

[10] Freund Y. and Schapire R.E. (1996). Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory.*

[11] Freund Y. and Schapire R.E. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, **14**(5): 771–780.

[12] Friedman J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, **29**(5): 1189–1232.

[13] Friedman J. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, **38**(4): 367–378.

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

[15] Gey S. and Poggi J. (2005). Boosting and instability for regression trees. *Computational Statistics and Data Analysis*, in press.

[16] Hastie T., Tibshirani R. and Friedman J. (2001). *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer–Verlag.

[17] Kearns M. and Valiant L. (1989). Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 433–444. ACM Press.

[18] Kearns M. and Vazirani U. (1994). *An Introduction to Computational Learning Theory.* MIT Press.

[19] Kutin S. and Niyogi P. (2001). The interaction of stability and weakness in AdaBoost. Technical Report TR-2001-30, University of Chicago.

[20] Nicholas T. (2003). Using AdaBoost and decision stumps to identify spam e–mail. Technical report, Stanford University, The Stanford Natural Language Processing Group.

`http://nlp.stanford.edu/courses/cs224n/2003/fp/tyronen/report.pdf`

[21] R Development Core Team (2005). R: *A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

`http://www.R-project.org`

[22] Ridgeway G. (1999). The state of boosting. *Computing Science and Statistics*, **31**: 172–181.

[23] Ridgeway G. (2005). `gbm`: *Generalized Boosted Regression Models.* R package version 1.5-1.

`http://www.i-pensieri.com/gregr/gbm.shtml`

[24] Ridgeway G. (2005). Generalized boosted models: A guide to the `gbm` package. Documentation for `gbm`.

[25] Rubin C., Schapire R. and Daubechies I. (2004). Boosting based on a smooth margin. In J. Shawe-Taylor and Y. Singer, eds., *Learning Theory: 17th Annual Conference on Learning Theory*, number 3120 in Lecture Notes in Computer Science, pages 502–517. COLT, Springer-Verlag.

[26] Schapire R.E. (1990). The strength of weak learnability. *Machine Learning*, **5**: 197–227.

[27] Schapire R.E. (1999). Theoretical views of boosting. In *Computational Learning Theory: Fourth European Conference.*

[28] Schapire R.E., Freund Y., Bartlett P. and Lee W. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics*, **26**(5): 1651–1686.

[29] Vuong A. (2003). Bagging and boosting regression trees. Master's Project, University of Calgary, Calgary AB, Canada.

[30] Wu Z., Hedges P. and Zhang S. (2005). Small business debt financing, family business, and discrimination: Evidence from a new Canadian survey. In *Proceedings of the 2005 ASAC Conference.*

[31] Zenko B., Todorovski L. and Dzeroski S. (2001). A comparison of stacking with MDTs to bagging, boosting, and other stacking methods. In *Proceedings of ECML/PKDD01 Workshop: Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pages 163–175. Albert-Ludwigs-Universität Freiburg, Germany.