8-11-2016

An Empirical Demonstration of the Probabilistic Upper Bound of the Adaptive Boosting Test Error

Paige Houston
pjh1123@rit.edu

ROCHESTER INSTITUTE OF TECHNOLOGY

MASTERS THESIS

An Empirical Demonstration of the Probabilistic Upper Bound of the Adaptive Boosting Test Error

Author:
Paige HOUSTON

Supervisor:
Dr. Ernest FOKOUÉ

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Applied Statistics

in the

School of Mathematical Sciences
College of Science

August 11, 2016

Declaration of Authorship

I, Paige HOUSTON, declare that this thesis titled, "An Empirical Demonstration of the Probabilistic Upper Bound of the Adaptive Boosting Test Error" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Committee Approval Form

Ernest Fokoué, Thesis Advisor, Associate Professor, School of Mathematical Sciences	Date
Steven LaLonde, Committee Member, Associate Professor, School of Mathematical Sciences	Date
Mei Nagappan, Committee Member, Assistant Professor, Golisano College of Computing and Information Sciences	Date
Daniel Lawrence, Honorary Committee Member, Professor, College of Engineering	Date

ROCHESTER INSTITUTE OF TECHNOLOGY

Abstract

Ernest Fokoué
School of Mathematical Sciences

Master of Science in Applied Statistics

An Empirical Demonstration of the Probabilistic Upper Bound of the Adaptive Boosting Test Error

by Paige HOUSTON

Statistical machine learning uses data to model a relationship between many parameters, or explanatory variables, and a response variable. The adaptive boosting algorithm is a machine learning method that can be used to model relationships of classification data. This method uses a weak base learner to improve accuracy of predicting the correct response class from a set of variables. Because of its learnability, adaptive boosting yields an exponentially decreasing empirical error. From this, an empirical error bound can be derived from the boosting algorithm. This empirical error bound inspires us to see if there is a generalized error bound and what form it takes. Evidence from boosting several real datasets will show that the generalized error follows the same shape as the empirical error, thus suggesting that a shift of the empirical error bound can create a generalized error bound. By simulating datasets from random and varying their characteristics based on criteria that seem to affect the shift, we can boost them and derive a function by which to shift the empirical error bound. We will record the test error of the boosted simulated datasets and build a regression model with that as the response and the varying characteristics of the datasets as the explanatory variables. The final regression model gives us the predicted outcome of the difference between the generalized error and the empirical error, thus enabling us to derive the suggested generalized error bound.

Contents

List of Figures

Chapter 1

Introduction

Statistical Machine Learning is used to construct a model or relationship between a set of parameters, or predictor variables, and a response to be able to predict results. First we must define a function space, $\mathscr{F}$, where our model lives. We never know the actual model, or relationship, between the predictors and the response, so it is crucial to define a function space. It is impossible to consider every function when looking for the "best" model, so by defining a function space, we can narrow down our search for the best model to just linear classifiers, polynomial classifiers, or kernels, etc. (Fokoue, 2015a).

The way we determine what is the "best" model is by analyzing the risk function. We essentially have two risk functions, the Theoretical Risk, or our Generalization Error, denoted by $R(f)$, and the Empirical Risk, or Training Error denoted by $\hat{R}_n(f)$ where f is a function in our function space $\mathscr{F}$, or $f \in \mathscr{F}$ (Fokoue, 2015a). The two functions are defined by the following:

$$R(f) = E[l(Y, f(X))]$$
$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n}[l(Y_i, f(X_i))],$$

where the function l is the loss function. The loss function can be defined as the "disagreement between the image $f(x)$ and the true image y" (Fokoue, 2015a, p.6). In other words, it takes into account if the predicted response, based on the model f, is the same as, or similar to, the actual response (Fokoue, 2015a).

The best possible function to model our data is denoted by f^*, and it is the one that minimizes the risk:

$$f^* = \underset{f}{\operatorname{argmin}}\{R(f)\}.$$

In practice, f^* cannot be found, so we estimate it using $\hat{f}_n$. Since we cannot simply choose a function with no background information that will happen to be close to our best function f^*, we, instead, look at all functions $f \in \mathscr{F}$, and choose the one that yields an empirical risk, $\hat{R}_n(f)$, closest to our theoretical risk, $R(f)$. However, we never know what the true error is in practice, so Vapnik and Chervonenkis derived an upper bound on the theoretical risk. This way we can know, generally, how large the error will be, instead of having no information on it at all. This theorem immediately follows in Section 1.1 (Fokoue, 2015a).

1.1 Vapnik and Chervonenkis Bound

Vapnik and Chervonenkis (1971) derived a bound for the theoretical risk. Their theorem is the following, as quoted directly from Fokoue, 2015a:

Theorem 1 (Vapnik and Chervonenkis Bound). *Let $\mathscr{F}$ be a class of functions implementing [some] learning machines, and let $\zeta = VCdim(\mathscr{F})$ be the VC dimension of $\mathscr{F}$. Let the theoretical and empirical risk be defined as earlier and consider any data distribution in the population of interest. $\forall f \in \mathscr{F}$, the prediction error (theoretical risk) is bounded by*

$$R(f) \leqslant \hat{R}_n(f) + \sqrt{\frac{\zeta(\log\frac{2n}{\zeta}+1) - \log\frac{\eta}{4}}{n}},$$

with probability of at least $1-\eta$. or

$$Pr\left\{ \texttt{TestError} \leqslant \texttt{TrainError} + \sqrt{\frac{\zeta(\log\frac{2n}{\zeta}+1) - \log\frac{\eta}{4}}{n}} \right\} \geqslant 1-\eta$$

In other words, it is a probabilistic bound, and depending on ones definition of η, the probability that our theoretical risk is bounded by the empirical risk plus a function of the dimension space and sample size is at least $1-\eta$.

A question we can ask is the following: Is there a more general takeaway from this theorem? Can we generate a probabilistic bound for the theoretical error in the form of the following equation:

$$R(f) \leqslant \hat{R}_n(f) + \phi(\ldots)$$

The above equation suggests that our theoretical risk can be bounded by a shift of the empirical risk by $\phi(\ldots)$ which is simply a function based on unknown parameters at this time.

Now we will revisit which function space, $\mathscr{F}$ to choose f from. As previously mentioned, $\mathscr{F}$ can be any of, but not limited to, the following:

- Linear Classifiers
- Kernel
- Ensembles
  - Bagging
  - Random Forest
  - Boosting

This list is just to give an idea of how many options there are, and this is not even a small fraction of all function spaces. However, in this thesis, we will focus on the Adaptive Boosting Algorithm, and how its generalized error can be bounded.

1.2 Adaptive Boosting

The Adaptive Boosting Algorithm is a classification method used in machine learning (Abney, Schapire, and Singer, 1999). The defined algorithm can be seen in Algorithm 1.

In this thesis, the response variable takes on values of either 0 or 1, meaning that a datum is classified in either Class 0 or Class 1. There are ways to adapt the algorithm and adjust the code to take on a multi-class response, or even multi-label data. By a multi-class response, we simply mean that there are more than two groups in the response, and by multi-label we mean that

Algorithm 1 The Adaptive Boosting Algorithm

```
1: **Let the following be defined as stated:**
2: Training data: $D_n = \{(x_i, y_i) : x_i \in X, y_i \in \{-1, +1\}, i = 1, 2, \ldots, n\}$.
3: Weak learners: $h_t(x)$ for $t = 1, 2, \ldots, T : X \to \{-1, +1\}$.
4: Error associated with the weak learners: $\epsilon_t$ for $t = 1, 2, \ldots, n$.
5: Initialized weights: $D_1(i) = \frac{1}{n}$ for $i = 1, 2, \ldots, n$.
6: Choose $h_t$ such that the error is minimized.
7: Choose $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$.
8: Update for $i = 1, 2, \ldots, n$,
       $$D_{t+1} = \frac{D_t(i)}{Z_t} * \begin{cases} e^{\alpha_t}, & \text{if } h_t(x_i) \neq y_i \\ e^{-\alpha_t}, & \text{if } h_t(x_i) = y_i \end{cases},$$
   meaning that
       $$D_{t+1} = \frac{D_t(i) \exp\big(-\alpha_t y_i h_t(x_i)\big)}{Z_t},$$
   where $Z_t$ is a normalization factor chosen so that $D_{t+1}$ is a distribution.
9: The final output is
       $$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$
```

at least one datum can be classified in more than one group (Schapire and Singer, 2000; Schapire and Singer, 1999; Abney, Schapire, and Singer, 1999). However, we will be using single-label datasets with a binary response of 0 or 1.

Note that the algorithm states that the response takes on values -1 and 1. Since most of the datasets we worked with came with a response that takes on values of 0 and 1, our adaptive boosting function in R converts them to -1 and 1, respectively, so the algorithm supports our data.

Our boosting algorithm uses unpruned *classification* trees as a base learner to build an adaptive model which essentially learns from its mistakes over the specified number of iterations. Throughout this thesis, unless otherwise specified, we will use a range for the number of iterations (or the number of trees boosted) from 1 to 50 (For more information on classification trees or regression trees, see Loh, 2011 or James et al., 2013).

1.2.1 The Algorithm

Before applying the adaptive boosting algorithm, we take a 60% stratified random sample on our dataset to get a training set. This means that we are taking 60% of our dataset to be in the training set, but we are intentionally taking 60% of the entries from Class 0 and 60% from Class 1 so we have the same proportions of the response class in the training set as we do in the original dataset. After the training set is determined, we apply the boosting algorithm. The algorithm used in this thesis begins by bootstrapping 75% of the training set. This means we are randomly selecting 75% of our dataset *with* replacement. Each datum has the same chance of being chosen to be in the subset during the first iteration. On that sample we will fit a classification model by, as previously stated, using unpruned classification trees as a base learner. When the model is built, we will use it to get a predicted class for all of the data in the original dataset. The first step is to make sure we have a decent base learner. What we call a "decent base learner" must predict the correct class of all the data points only slightly better than if we randomly guessed each class. In other words, if the predicted proportion of misclassification (error $= \epsilon_t$) is just under 0.5, then it is a decent base learner (Ohio, 2015; Fokoue, 2015b; Quinlan, 2006). In the case that it is not a decent base learner, the algorithm repeats the bootstrapped sample and rebuilds a model until it meets the decent base learner criteria (Ohio, 2015).

In the case that it is a decent base learner, or once the resampled data yields a decent base learner, the Boosting Algorithm recalculates the weights associated with each data point. The updated weights are based on the *misclassified* points from that iteration. First, recall that in the first iteration each data point has an equal chance of being selected in the bootstrapped sample. This is because the weights are defaulted to $\frac{1}{n}$ (See Algorithm 1, line 7). Now, if there were a total of 1000 data points and after the first iteration 400 points were misclassified, the sum of the weights on the misclassified points would be $\epsilon_1 = \frac{400}{1000}$. The weight of the data points for the next iteration begins getting updated by taking ϵ_1 to calculate a value α_t associated with each data point (Ohio, 2015). The value of α_t is determined by

$$\alpha_t = \frac{1}{2} log\Big(\frac{1-\epsilon_t}{\epsilon_t}\Big).$$

It is important to note that we know α_t is always going to be greater than 0 because the algorithm does not continue on unless it is better than random guessing, meaning $\epsilon_t < \frac{1}{2}$. As long as $\epsilon_t < \frac{1}{2}$, we can easily see that α_t will be greater than 0 (Ohio, 2015).

Next, a new weight is calculated and applied to the data points by the following equation:

$$w_{t+1} = w_t e^{\alpha_t I(h_t(x) \neq y)}.$$

The value of $I(h_t(x) \neq y)$ is 0 when the point is correctly classified and 1 when it is not, so it is important to note that the correctly classified data points *do not* get an updated weight. When a point is correctly classified, the exponential term in this equation becomes 1 which, in turn, outputs the weight of the previous iteration, $w_{t+1} = w_t$. On the other hand, the misclassified data points get a larger weight so that they have a greater chance of being selected for the next iteration. There is a restriction on these weights, however. The weights must sum to 1, so that a consistently misclassified point's weight does not begin growing out of bound. Thus, after the new weights are assigned, each one gets divided by the sum of all values w_t for $t = 1, 2, \ldots, T$, and that value becomes the new associated weight to the data points for that next iteration. In reference to Algorithm 1, $\sum_{t=1}^{T} w_t = Z_t$, so the new weights assigned are:

$$D_{t+1} = \frac{D_t(i)\exp\big(-\alpha_t y_i h_t(x_i)\big)}{Z_t}.$$

Then another bootstrapped sample takes place based on these weights, and the entire loop repeats. This happens a chosen number of times, T, which, again, we are testing in the form of a range from 1 to 50 (Ohio, 2015).

During each of the T iterations, the base model and the α values for that model are stored. When using the boosting algorithm to predict classification, the list of α values act as a weight for each model. What this means is the lower the ϵ for each model, the higher the weight, α. In the case of predicting one point, the boosting algorithm takes a weighted majority of the T iterations using α as the weight for each model. Predicting this way makes the results much more accurate because the models with a smaller error have a higher weight on the final outcome. Also, because this model essentially learns from its mistakes, it is not surprising that the training error will continually decrease as the iterations carry on. This introduces the idea of a bound on the training error.

1.2.2 The Boosting Function in R

The analysis for this thesis is done in The R Project for Statistical Computing R, 2016. We have two options for a boosting function inside R. The first option is to use the `boosting()` function in the "adabag" package inside of R (Alfaro-Cortes et al., 2016; Alfaro et al., 2015). Second, we have a handwritten boosting function (Fokoue, 2015b). This code was written in an attempt to easily alter the base learner so error from different base learners could be compared. Fokoue, 2015b's function is called `boosted.trees()`, which can be found in Appendix A. When these two functions run side by side, they yield extremely similar results. The `boosting()` function in R takes much longer to run; therefore, we are choosing to use Fokoue, 2015b's function instead. Margineantu and Dietterich, 1997 state that ensemble methods, such as Adaptive Boosting, require a lot of memory on a computer (Refer to their report to see the exact number of Kbytes needed per a specific amount of trees boosted (Margineantu and Dietterich, 1997)). Since this is evidence that the required analysis for this thesis could take an incredibly long amount of time and memory, it supports our choice to use Fokoue, 2015b's boosting function over the one in the adabag package (Alfaro-Cortes et al., 2016; Alfaro et al., 2015).

We ran $T = 50$ iterations of adaptive boosting using each function on the dataset Ionosphere, and saved the test error that each method generated (Sigillito, 1989). The results were nearly always the same or close. In fact, the mean difference between the two functions over each iteration is 0.00014. The average test error results of the 50 iterations for the R function and Fokoue, 2015b's written function were 0.06964539 and 0.06950355, respectively. Also, Figure 1.1 shows the distribution of the training error from all iterations of each boosting function with the number of base trees set to 50. Again, since they yield very similar results, and the `boosted.trees()` function is more efficient, we have made the decision to use that boosting function (Fokoue, 2015b).

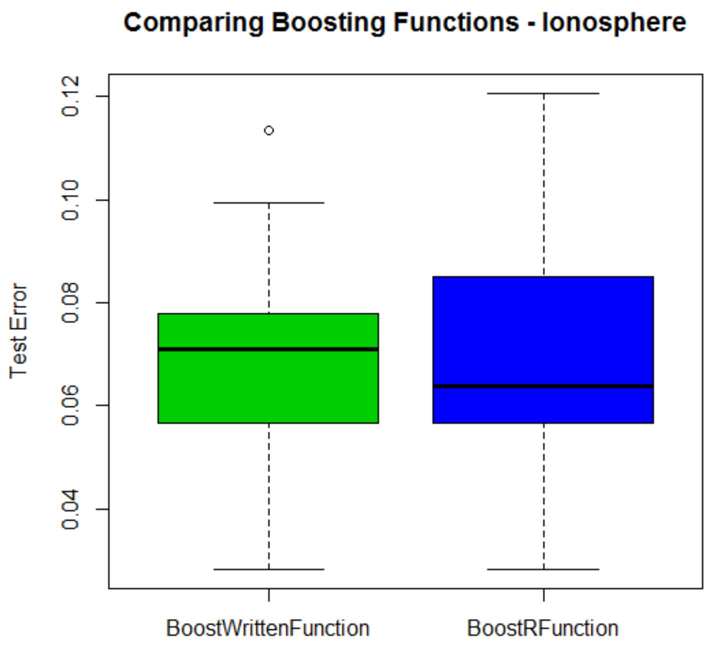

FIGURE 1.1: Training Error from Both Boosting Functions

1.3 The Main Focus

In Chapter 2, we will explain and prove the Training Error Bound Theorem in terms of Schapire and Freund, 2012, which is characterized in the form of:

$$\Pr_{i\sim D_1}[H(x_i) \neq y_i] \leqslant \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \leqslant exp\Bigg(-2\sum_{t=1}^{T}\gamma_t^2\Bigg).$$

where $\Pr_{i\sim D_1}[H(x_i) \neq y_i]$ is the specific empirical risk function for the boosting algorithm equal to $R_1(H)$ and H is the chosen function/model.

The fact that there exists a training error bound prompts the following question: What does the generalized error look like, and how can we model it? Our empirical training and test error of various real datasets will suggest there is a relationship between the training error and test error. More specifically, there seems to be a shift between the two. The main problem in this thesis is determining how we will model the test error. Based on the empirical error results, we will simulate datasets from random that vary in seemingly important characteristics to see exactly how each of these characteristics impacts the test error. Some of the characteristics we are speaking of are the sample size, the number of parameters, and their ratio. After boosting the several simulated datasets, we will solve the main problem in this thesis by performing Multiple Linear Regression to determine the relationship between the characteristics of each simulated dataset and that dataset's average test error yielded by the adaptive boosting algorithm. This will enable us to model the test error of datasets based on their specific sizes, dimensions, and characteristics.

Chapter 2

Error Analysis for Adaptive Boosting

2.1 Training Error Bound

Adaptive Boosting is known to improve data quality because the algorithm was strategically created to choose the more difficult data points to be in the training set. As discussed previously, the points associated with a larger weight are the points that are continually misclassified, so, as a result, those points have a greater chance of being selected to be in the training set (Abney, Schapire, and Singer, 1999). Because of this, the data becomes trained rather quickly, without compromising on accuracy (Appel et al., 2013; Drucker, Schapire, and Simard, 1993). This brings us to the training error bound.

Many researchers discuss the training error bounds of the boosting algorithm (Schapire and Freund, 2012; Ohio, 2015; Freund and Schapire, 1996; Drucker and Cortes, 1996). An "informal" way of thinking about the training error bound is if the final classification (based on the majority weight) is incorrect, this would mean that the majority of the previous iterations would have also misclassified this data point (Schapire and Freund, 2012, p. 54). Since it was consistently misclassified, the weight on this point would be increasingly large. This is because, during boosting, points get an increased weight for each iteration that they are misclassified. Since the sum of the weights of all the points is 1, by definition, the number of data points with extremely high weights would have to be limited. Thus, the number of misclassified points will be decreasing with each iteration, meaning the training error will be getting closer and closer to 0 (Schapire and Freund, 2012).

The Training Bound for Boosting Theorem, as stated by Schapire and Freund, 2012, is the following:

Theorem 2 (Training Error Bound for Boosting). *Given the notation for the boosting algorithm stated in Algorithm 1 and Section 1.2.1, let* $\gamma_t \doteq \frac{1}{2} - \epsilon_t$*, and let* D_1 *be an arbitrary initial distribution over the training set. Then the weighted training error of the combined classifier H with respect to* D_1 *is bounded as:*

$$\Pr_{i \sim D_1}[H(x_i) \neq y_i] \leqslant \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leqslant exp\Bigg(-2 \sum_{t=1}^{T} \gamma_t^2 \Bigg).$$

Note: γ_t *is always positive because* $\epsilon_t \leqslant \frac{1}{2}$.

Proof. Recall the following are defined as:

- Training data: $D_n = \{(x_i, y_i) : x_i \epsilon X, y_i \epsilon \{-1, +1\}, i = 1, 2, \ldots, n\}$.
- Weak learners: $h_t(x)$ for $t = 1, 2, \ldots, T : X \rightarrow \{-1, +1\}$.
- Initialized weights: $D_1(i) = \frac{1}{n}$ for $i = 1, 2, \ldots, n$.
- Choose h_t such that the error is minimized.
- Choose $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$.
- Update for $i = 1, 2, \ldots, n$,

$$D_{t+1} = \frac{D_t(i)}{Z_t} * \begin{cases} e^{\alpha_t}, & \text{if } h_t(x_i) = y_i \\ e^{-\alpha_t}, & \text{if } h_t(x_i) \neq y_i \end{cases},$$

  meaning that

$$D_{t+1} = \frac{D_t(i) \exp\big(-\alpha_t y_i h_t(x_i)\big)}{Z_t},$$

  where Z_t is a normalization factor chosen so that D_{t+1} is a distribution. The final output is

$$H(x) = \text{sign}\Bigg(\sum_{t=1}^{T} \alpha_t h_t(x) \Bigg).$$

- Let $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

Expanding the equation for D_{T+1} over the occurrences up to $T+1$ in terms of D_t, we get:

$$D_{T+1}(i) = D_1(i) \times \frac{\exp\big(-\alpha_1 y_i h_1(x_i)\big)}{Z_1} \times \frac{\exp\big(-\alpha_2 y_i h_2(x_i)\big)}{Z_2} \times \dots$$

$$\dots \times \frac{\exp\big(-\alpha_T y_i h_T(x_i)\big)}{Z_T}$$

$$= \frac{D_1(i)\exp\big(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\big)}{\prod_{t=1}^{T} Z_t} = \frac{D_1(i)\exp\big(-y_i F(x_i)\big)}{\prod_{t=1}^{T} Z_t}.$$

Recall that $H(x) = sign\big(F(x)\big)$.

So, if $H(x) \neq y$, then $yF(x) \leqslant 0$.

This means that $e^{-yF(x)} \geqslant 1$.

It follows that $1\{H(X) \neq y\} \leqslant e^{-yF(x)}$,

meaning 1 is less than the exponential of $-yF(x)$, given that the response is wrongly classified.
Now, the weighted training error becomes:

$$\Pr_{i \sim D_1}[H(x_i) \neq y_i] = \sum_{i=1}^{n} D_1(i) 1\{H(x_i) \neq y_i\} \leqslant \sum_{i=1}^{n} D_1(i)\exp\big(-y_i F(x_i)\big)$$

$$= \sum_{i=1}^{n} D_{T+1}(i) \prod_{t=1}^{T} Z_t.$$

Since D_{T+1} is a distribution, $\sum_{i=1}^{n} D_{T+1}(i) = 1$.
Thus,

$$\sum_{i=1}^{n} D_{T+1}(i) \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t.$$

Since $D_{t+1} = \frac{D_t(i)\exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$ and Z_t is the normalizing factor, Z_t must make the following true:

$$\frac{\sum_{i=1}^{n} D_t(i)\exp\big(-\alpha_t y_i h_t(x_i)\big)}{z_t} = 1.$$

Therefore,

$$Z_t = \sum_{i=1}^{n} D_t(i)\exp\big(-\alpha_t y_i h_t(x_i)\big).$$

Since both y_i and $h_t(x_i)$ take on the values $\{-1,+1\}$, when h_t misclassifies, $y_i h_t(x_i) = -1$, and when it correctly classifies, $y_i h_t(x_i) = +1$. Therefore, we can break the above equation down to:

$$z_t = \sum_{i:y_i=h_i(x_i)} D_t(i)e^{-\alpha_t} + \sum_{i:y_i\neq h_i(x_i)} D_t(i)e^{\alpha_t}.$$

The second part of this equation can be recognized as the error multiplied by e^{α_t}, and the first part is 1 minus the error multiplied by $e^{-\alpha_t}$, so it simplifies to:

$$z_t = e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t.$$

And we know that $\epsilon_t = \frac{1}{2} - \gamma_t$, so it follows that

$$z_t = e^{-\alpha_t}\Big(\frac{1}{2}+\gamma_t\Big) + e^{\alpha_t}\Big(\frac{1}{2}-\gamma_t\Big).$$

Since we are strategically taking $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$, the above equation simplifies to:

$$e^{-\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}}\Big(\frac{1}{2}+\gamma_t\Big) + e^{\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}}\Big(\frac{1}{2}-\gamma_t\Big).$$

Again, from the fact that $\epsilon_t = \frac{1}{2} - \gamma_t$, we can write

$$\begin{aligned} z_t &= \left(\frac{\frac{1}{2}+\gamma_t}{\frac{1}{2}-\gamma_t}\right)^{-\frac{1}{2}}\Big(\frac{1}{2}+\gamma_t\Big) + \left(\frac{\frac{1}{2}+\gamma_t}{\frac{1}{2}-\gamma_t}\right)^{\frac{1}{2}}\Big(\frac{1}{2}-\gamma_t\Big) \\ &= \left(\frac{\frac{1}{2}-\gamma_t}{\frac{1}{2}+\gamma_t}\right)^{\frac{1}{2}}\Big(\frac{1}{2}+\gamma_t\Big) + \left(\frac{\frac{1}{2}+\gamma_t}{\frac{1}{2}-\gamma_t}\right)^{\frac{1}{2}}\Big(\frac{1}{2}-\gamma_t\Big) \end{aligned}$$

$$= \left(\frac{1}{2}-\gamma_t\right)^{\frac{1}{2}}\left(\frac{1}{2}+\gamma_t\right)^{\frac{1}{2}}+\left(\frac{1}{2}+\gamma_t\right)^{\frac{1}{2}}\left(\frac{1}{2}-\gamma_t\right)^{\frac{1}{2}}$$

$$= \left[\left(\frac{1}{2}-\gamma_t\right)\left(\frac{1}{2}+\gamma_t\right)\right]^{\frac{1}{2}}+\left[\left(\frac{1}{2}+\gamma_t\right)\left(\frac{1}{2}-\gamma_t\right)\right]^{\frac{1}{2}} = 2\left[\left(\frac{1}{2}-\gamma_t\right)\left(\frac{1}{2}+\gamma_t\right)\right]^{\frac{1}{2}}$$

$$2\left(\frac{1}{4}-\gamma_t^2\right)^{\frac{1}{2}} = 2\sqrt{\frac{1}{4}-\gamma_t^2} = \sqrt{4(\frac{1}{4}-\gamma_t^2)} = \sqrt{1-4\gamma_t^2}.$$

Now, we have proven that $\Pr_{i\sim D_1}[H(x_i)\neq y_i] \leqslant \prod_{t=1}^{T}\sqrt{1-4\gamma_t^2}$.

Next, we must prove that

$$\Pr_{i\sim D_1}[H(x_i)\neq y_i] \leqslant \prod_{t=1}^{T}\sqrt{1-4\gamma_t^2} \leqslant \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right), \text{ for any} \gamma > 0.$$

We start by applying the following approximation:

$$1+x \leqslant e^x \text{for all real } x.$$

We must show that the condition:

$$\epsilon_t \leqslant \frac{1}{2}-\gamma \text{ for } \gamma > 0 \text{ for each } t^{th} \text{ round },$$

implies the following inequality:

$$\sqrt{1-4\gamma_t^2}^{T} \leqslant e^{-2\gamma^2 T}.$$

Since we know that $1+x \leqslant e^x$, if we set $x=-2\gamma_t^2$, then for each γ_i for $i \,\epsilon\, \{1,2,\dots,T\}$, it follows that $1-2\gamma_t^2 \leqslant e^{-2\gamma_t^2}$. This holds because all $\gamma_i \in \mathbb{R}$, for $i \,\epsilon\, \{1,2,\dots,T\}$. Now, since we are dealing with positive quantities, we have the following:

$$\sqrt{(1-2\gamma_t^2)^2} \leqslant e^{-2\gamma_t^2} \rightarrow \sqrt{1-4\gamma_t^2+4\gamma_t^4} \leqslant e^{-2\gamma_t^2}.$$

Since $\gamma_t > 0$, we can say that

$$\sqrt{1-4\gamma_t^2} \leqslant \sqrt{1-4\gamma_t^2+4\gamma_t^4} \leqslant e^{-2\gamma_t^2}.$$

Therefore, $\sqrt{1-4\gamma_t^2} \leqslant e^{-2\gamma_t^2}$.

Since this holds true for each individual γ_i, it also holds for the sums over all $i \epsilon 1, 2, \ldots, T$.

Hence, we can write $\prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \leqslant \prod_{t=1}^{T} e^{-2\gamma_t^2}$.

Lastly, this simplifies to $\prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \leqslant e^{(-2\sum_{t=1}^{T}\gamma_t^2)}$.

We can conclude, then, that

$$\Pr_{i\sim D_1}[H(x_i) \neq y_i] \leqslant \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \leqslant \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

(Schapire and Freund, 2012). □

We can see now that the training error bound has a decreasing exponential shape. Should we expect the generalized error to take that same form? In Section 2.2, we will show the boosting test and training error plots for 10 real datasets. These plots will suggest that the generalized error *does* have the same decreasing exponential form as the training error.

2.2 Empirical Demonstration of the Training Error

Using 10 datasets of various dimensions, we will run 100 iterations of the boosting function to have an empirical look at how the test and training error behave when ranging the number of base trees from 1 to 50. The 10 datasets analyzed are the following:

- Ionosphere (Sigillito, 1989)
- Breast Cancer (Lichman, 2013b)
- Crash (Lucas et al., 2013)

- Dermatology (Lichman, 2013a)
- Letter Recognition (Slate, 1998)
- Colon Cancer (Alon et al., 1999)
- Lung Cancer (Institute, 2016)
- Lymphoma (Fokoue, 2016a)
- Lymphoma 2 (Fokoue, 2016a)
- Prostate Cancer (Fokoue, 2016b)

The lymphoma dataset was subset in two separate ways to create two datasets, which we call Lymphoma, and Lymphoma 2.

The first 5 datasets on the list are ones with the sample size larger than the number of parameters, denoted by $n > p$, and the second 5 datasets are ones with a larger number of parameters than sample size, denoted by $p > n$ (see Table 2.1).

Table 2.1 shows the dimensions of each dataset analyzed, as well as the balance of the response classes. The reason we chose to note these characteristics of the datasets is because datasets with a large number of parameters and small sample sizes tend to be more unstable, thus yielding a larger error. Also, Mazurowskia et al., 2008 found that the error tends to increase when the classes are imbalanced in the training set. What we mean by class imbalance is the response in the training set having more data classified as one group over another. An example of this would be if a training set had more data being classified as 0 than 1. In this case, the model will have more information on Class 0, and the out-of-sample predictions might generate more Class 0 than Class 1. If the actual response happens to be more commonly Class 1, then there will be a higher rate of misclassification, and thus a higher test error (Mazurowskia et al., 2008) (For more information on addressing the issue of imbalanced classes, see Longadge, Dongre, and Malik, 2013).

Now we can analyze the error results from boosting these 10 datasets. More specifically, we will look to see what shape the generalized error has.

Dataset	n	p	Ratio n/p	Class 0	Class 1	Ratio 0/1
Ionosphere	351	32	10.97	126	225	0.56
Breast Cancer	699	10	69.90	241	458	0.53
Crash	540	18	30.00	46	494	0.09
Dermatology	358	34	10.53	247	111	2.23
Letter Recognition	1573	16	98.31	786	787	1.00
Colon Cancer	62	2000	0.03	40	22	1.82
Lung Cancer	197	1000	0.20	139	58	2.40
Lymphoma	180	621	0.29	93	87	1.07
Lymphoma 2	93	621	0.15	51	42	1.21
Prostate Cancer	79	500	0.16	37	42	0.88

TABLE 2.1: Dimensions and Class Balance/Imbalance for the Various Datasets

Does it have the same decreasing exponential form as the training error? The error results for the 10 datasets are shown in Figures 2.1 and 2.2.

The first noticeable aspect of the results is that the error flattens out relatively quickly in all cases. What this means for the boosting algorithm is that we do not need to select a large number of trees to boost to get an accurate result. In both Figures 2.1 and 2.2, we can see the training and test error begin to flatten around $T = 20$ trees. This means that whether we boost 20 or 50 trees, we are going to get very similar error results. It is better to use fewer trees for the sake of memory usage and efficiency. On the other hand, if one wanted to be as accurate as possible and was not worried about the amount of time it takes the code to run, he or she could use 50 trees because, with boosting, we do not have to be concerned about over fitting our data (Margineantu and Dietterich, 1997; Schapire, 1999).

Next, it is interesting to note that the test error *does* have a similar decreasing exponential shape as the training error (Schapire and Singer, 1999). This is evidence that we will be able to model a generalized bound similar to the Vapnik and Chervonenkis bound in Theorem 1. We are speculating that our bound will be of the form:

$$R(H) \leqslant \hat{R}_n(H) + \phi(\ldots)$$

What we do not know, though, is what ϕ is. We are modeling ϕ to be the difference between the theoretical risk and empirical risk. Since the empirical risk, or training error, plateaus at 0, all we must do to model the difference, is model the test error. That brings us to the question: What causes the test error to be larger in some datasets in comparison to others? In Chapter 3, we will share some conjectures inspired by the empirical demonstrations that may answer our question.

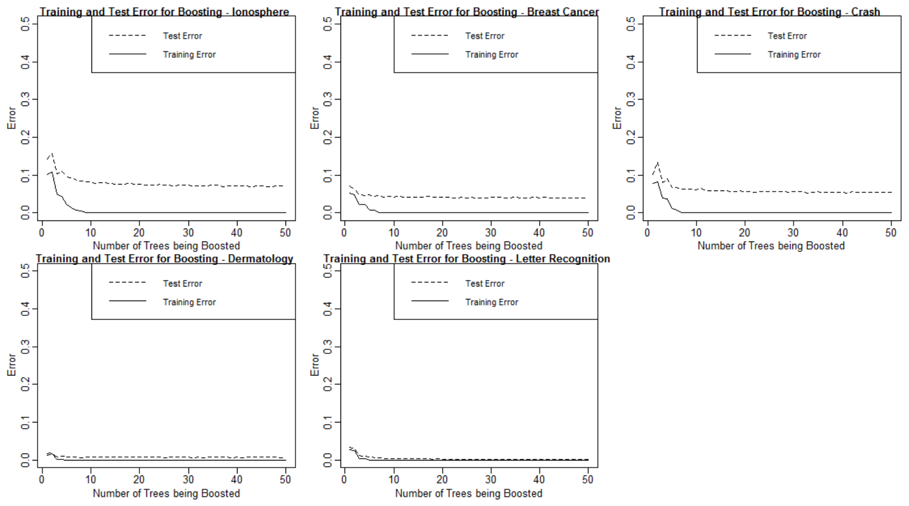

FIGURE 2.1: Training and Test Error on Large n Small p Datasets

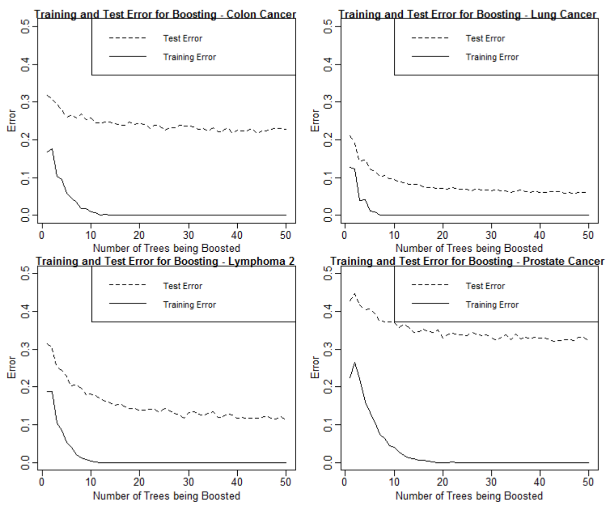

FIGURE 2.2: Training and Test Error on Large p Small n Datasets

2.3 Existing Bounds

In this section we will state some existing generalized error bounds that are highly sophisticated and quite technical. First, we have a generalized error bound from Schapire and Ene, 2006. This bound is as follows:

$$R(H) \leqslant \hat{R}_n(H) + O\left(\sqrt{\frac{T\log|H| + T\log\frac{n}{T} + \log\frac{1}{\eta}}{n}}\right).$$

In this equation, H, T, and n are defined as in Algorithm 1, and η is defined as in Theorem 1. This bound is quite involved and contains many different factors (For more information on this bound, see Schapire and Ene, 2006).

The next bound is one by Vaughan et al., 2006. This bound is defined as:

$$R(H) \leqslant \hat{R}_n(H) + \tilde{O}\left(\sqrt{\frac{T\zeta}{n}}\right).$$

Here, H, T, and n are also defined as in Algorithm 1, and ζ is defined as in Theorem 1. The term $\tilde{O}(.)$ denotes that the $\log$ terms are removed from that part of the equation (See Vaughan et al., 2006 for more information on this bound).

Lastly, Koltchinskii, Panchenko, and Lozano, 2001 have a paper on a generalized error bound, as well. This bound is the most complex of the bounds listed in this section. Koltchinskii, Panchenko, and Lozano, 2001 derive bounds using the VC dimension, and other calculated constants that may be difficult to comprehend (For more information on this bound and to see it stated explicitly, see Koltchinskii, Panchenko, and Lozano, 2001, Section 2).

These and other generalized error bounds are very complex and beyond the scope of this thesis. Accordingly, we will move on with our research to find a more functional bound, with easier terms to understand.

Chapter 3

Empirical Demonstration of the form of Generalization Error of Boosting

3.1 Conjectures Gathered from Real Datasets

In reference to Figures 2.1 and 2.2, we can now discuss conjectures we have about which characteristics of our dataset, if any, are affecting our test error results. First, we note, in general, how much larger the test error is in the cases with large p small n datasets versus the small p large n datasets. Since all of the graphs are plotted on the same scale, we can easily see that the test error in Figure 2.2 is generally greater than that in Figure 2.1. The datasets in Figure 2.2 that stand out are the Colon Cancer and Prostate Cancer datasets (Alon et al., 1999; Fokoue, 2016b). These two dataset yield the largest test error out of all of the datasets tested. Referring back to Table 2.1, we can see that the Colon Cancer and Prostate Cancer datasets have the smallest sample sizes out of all of the datasets at 62 and 79, respectively. On the contrary, looking at Figure 2.1 we can see that the Letter Recognition dataset clearly has the smallest test error (Slate, 1998). This is the dataset with the largest sample size that was tested, at 1573 observations, it is more than double the sample size of the second largest dataset tested.

Based on these three examples, it would appear that sample size must have a huge effect on the test error. Whether or not this conjecture is correct, the Dermatology dataset, which can be seen in Figure 2.1, yields counter intuitive results (Lichman, 2013a). This dataset shows the second lowest test error to the Letter Recognition dataset. From the previous conjecture, we would assume that the sample size for the Dermatology dataset must be relatively large, but in reality, it is not. This dataset has 358 observations, one of the smallest sample sizes in the $n > p$ category. From our general results, it appears that sample size is an important factor in what affects test error, but this leads us to believe something else must also play a crucial role.

Perhaps the number of parameters could also affect the test error. We took subsets of the Lymphoma dataset to create two different $p > n$ datasets. We can then directly compare the results of two datasets since they have the same number of parameters but different sample sizes (Fokoue, 2016a). (Note: We may only be able to generalize this to $p > n$ datasets). From Figure 2.2, it seems as though the Lymphoma 2 dataset yields a larger error than the first Lymphoma dataset. The only difference in dimensions is the sample size. Lymphoma 2 has a sample size of 93, whereas Lymphoma (yielding a smaller test error) has a sample size of 180, almost double that of Lymphoma 2. Also, as previously stated, the Colon Cancer Dataset and Prostate Cancer dataset both have extremely small sample sizes, but their number of parameters differ greatly with Colon Cancer having 2000 parameters and Prostate Cancer having 500. Although both datasets yield a relatively large test error, the Prostate Cancer dataset's test error is evidently larger than the Colon Cancer dataset's test error. This may suggest that the greater the number of parameters, the lower the error. We cannot make any conclusions just yet, but this is a good place to start.

Next, we will consider the class balance versus imbalance. Here, we will focus on the column titled "Ratio 0/1" in Table 2.1. As previously stated, it could be assumed that with a more balanced class ratio (i.e. with the ratio closer to 1), the test error would be minimized. If we have equal information on both classes, it is easier to correctly classify a test set, even if the majority of the test set is truly one group over the other. In Table 2.1, it is readily apparent that the Letter Recognition dataset has almost exactly balanced data. As mentioned previously, this dataset yields the smallest test error. We were originally attributing its low test error to the large sample size, but perhaps it is due to the balanced classes of the response. The Lymphoma dataset also has a 0 to 1 ratio close to 1 and yields one of the lowest test error of the $p > n$ datasets. The Lymphoma 2 and Prostate Cancer datasets both have a similar ratio of slight imbalance (with one slightly over 1 and the other slightly under), yet they produce very different test error results. This leads us to believe that if class imbalance is important, its results may vary based on the previous factors mentioned, or even the ratio of n to p, which is also displayed in Table 2.1. Lastly, the ratio of Class 0 to Class 1 in the Dermatology dataset and the Prostate Cancer dataset are very similar at 2.23 and 2.40, respectively. Their results are also interesting because, even though the Dermatology dataset has $n > p$ and the Prostate Cancer dataset has $p > n$, they both yield some of the lowest test error results of the other 4 datasets in their respective categories.

In conclusion, it is apparent that sample size, number of parameters, and class imbalance are important factors in the test error outcome but how and in what way, cannot be defined in a simple rule that applies to all datasets. This does, however, give us enough information to simulate multiple datasets of varying sizes for the purpose of applying the boosting algorithm in the same matter as earlier. Since we have evidence that the generalized error of the adaptive boosting algorithm will have the same decreasing exponential shape as the training error, we will derive a multiple linear regression model, ϕ, using these mentioned factors as the parameters with the averaged test error as the response. The function ϕ will then be added to our empirical risk to create a theoretical risk bound similar to the VC Bound in Theorem 1.

3.2 Simulated Datasets

Since the existing bounds discussed in Section 2.3 are too technical and sophisticated for use in this study, we will go on to create strategically simulated datasets of various characteristics that the empirical demonstration in Figure 2.1, Figure 2.2, and Section 3.1 showed may be affecting the generalized error. Simulated datasets are a good tool to use when analyzing a theory or conjecture. In the case of simulated datasets, we know the truth because we created the datasets, they are available in any quantity, and we know the format will be exactly how we want it (*Observational Medical Outcomes Partnerships* 2013). There are multiple ways to simulate datasets in R. Melnykov, Chen, and Maitra, 2012a use the R function called `simdataset()` from the `MixSim` package to simulate datasets (Melnykov, Chen, and Maitra, 2012b; Melnykov et al., 2015). Another helpful tool for simulating datasets is the `sim.item()` function from the `psych` package in R (Revelle, 2006; Revelle, 2015). However, we are using a function created by Ramos, 2016 called `simclass()`. This function enables us to generate datasets of any dimensions. The underlying relationship between the variables and the response can be seen within the function in Appendix A. We kept this underlying relationship constant for all of the simulated datasets. This relationship could effect our results, so by keeping it constant here, we are able to attribute the changes in our results to only the characteristics we are purposly changing. Ramos, 2016's function uses the calculated response as a probability in the random binomial R function, `rbinom()` to make the response either a 1 or 0 to give us our binary response (R, 1993b).

The idea in this chapter is to take what we learned from our empirical test error results and simulate datasets to test our conjectures. We will study a range of:

- Sample size
- Number of parameters
- Class imbalance/balance

to strategically construct simulated datasets. We will apply the adaptive boosting algorithm to each dataset, ranging the number of trees from 1 to 50 and replicate that 100 times. The results can be seen in Section 3.3, where there is an analysis of the test and training error plots.

3.2.1 Creating Imbalanced Simulated Datasets

We began by using Ramos, 2016's function called `simclass()` and constructed 5 datasets of each of the varying sizes and dimensions given by:

- $n = 250, p = 17$
- $n = 80, p = 17$
- $n = 250, p = 1200$

- $n = 80, p = 1200$

The largest sample size we chose to work with is 250, while the smallest is 80. We did not want to construct datasets that were much too large for the machine to handle, and a sample size of 250 compared to 80 should show the contrast in results that we are looking for without it being unmanageable. Also, the largest number of parameters we chose to simulate is 1200 while the smallest is 17. This is a large range and seems to be realistic based on the real datasets we were working with. Since we simulated 5 of each size and dimension, we now have 20 datasets, and as seen from the list, 10 of these datasets are of the type small p large n, and the other 10 data sets are of the type large p small n.

It became apparent that, by default, this function yields an imbalanced class result, meaning the number of data points in Class 0 was much smaller than the number of data points in Class 1. In fact, the ratio of data points in our datasets classified as Class 0 to those classified as Class 1 ranged anywhere from 0.20 to 0.60. One conjecture discussed is that the ratio of data points in Class 0 to those in Class 1 could alter our test error results, so we altered the function to create datasets with a balanced response, as well (i.e., the ratio of Class 0 to Class 1 is exactly 1). This function, called `simclasseven()`, can also be found in Appendix A. We then used this altered function to create 20 more datasets of the same varying size and dimension as the previous 20 datasets.

3.2.2 Creating Balanced Simulated Datasets

The way we generated `simclasseven()` came directly from Ramos, 2016's simulation function `simclass()`. The reason `simclass()` was returning imbalanced results is because it uses the R function `rbinom()` to calculate the response of our simulated data (R, 1993b; Ramos, 2016). We altered the code to use the R function `median()` to calculate the response (R, 1993a). This way, the function takes the calculated relationship between the variables and the response `pii` and classifies anything below the median of `pii` as 0 and anything above the median as 1. Now, we have a balanced response every time.

Next, we will apply the adaptive boosting algorithm to each of the 40 simulated datasets, ranging the number of trees from 1 to 50 and replicating that 100 times. The results can be seen in Figures 3.1 through 3.8, and the trends will be discussed in Section 3.3.

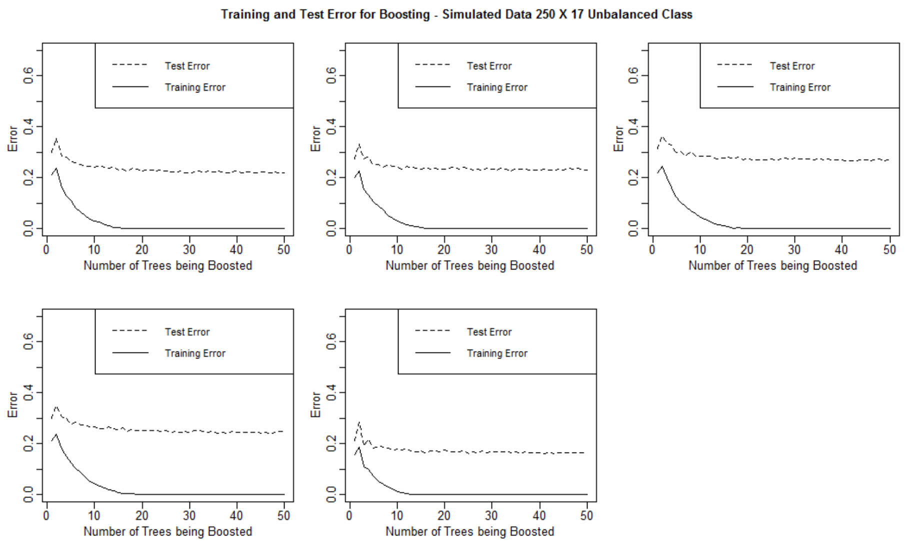

FIGURE 3.1: Training and Test Error on Imbalanced Datasets of Size 250 X 17

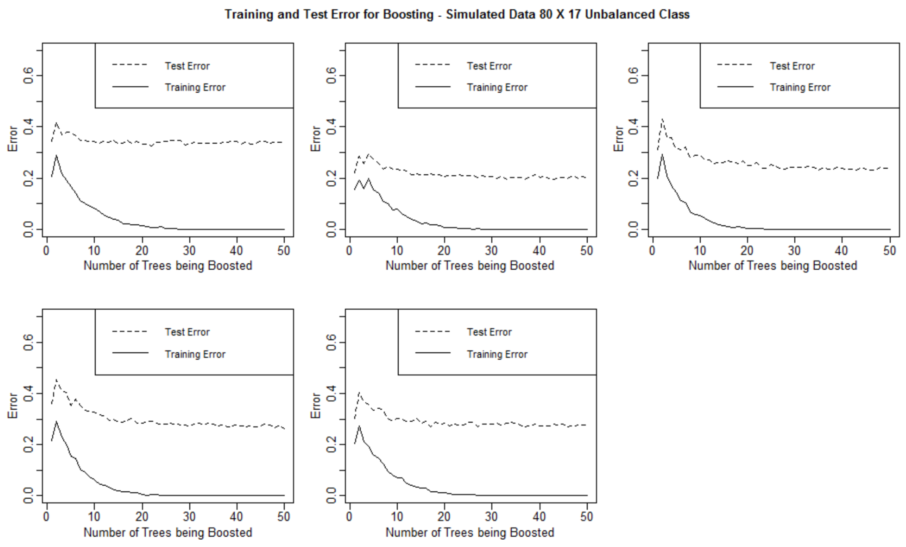

FIGURE 3.2: Training and Test Error on Imbalanced Datasets of Size 80 X 17

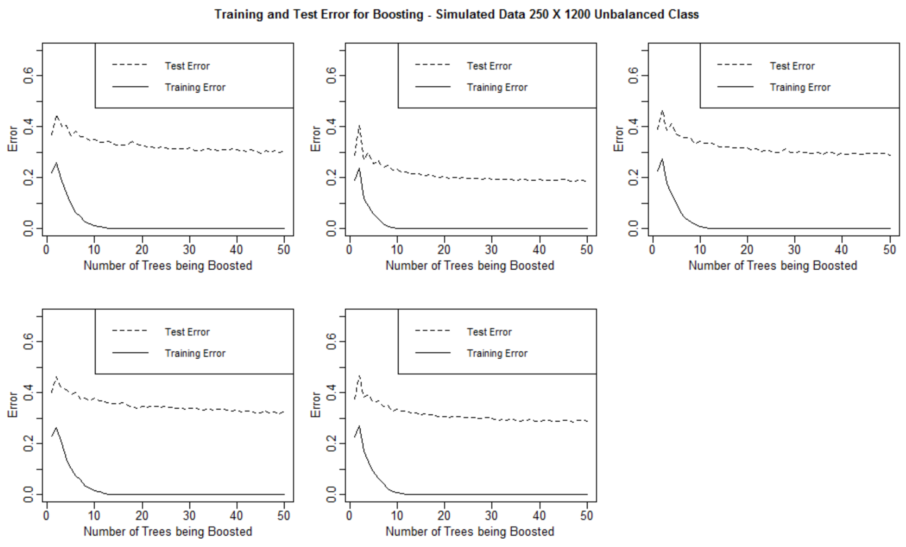

FIGURE 3.3: Training and Test Error on Imbalanced Datasets of Size 250 X 1200

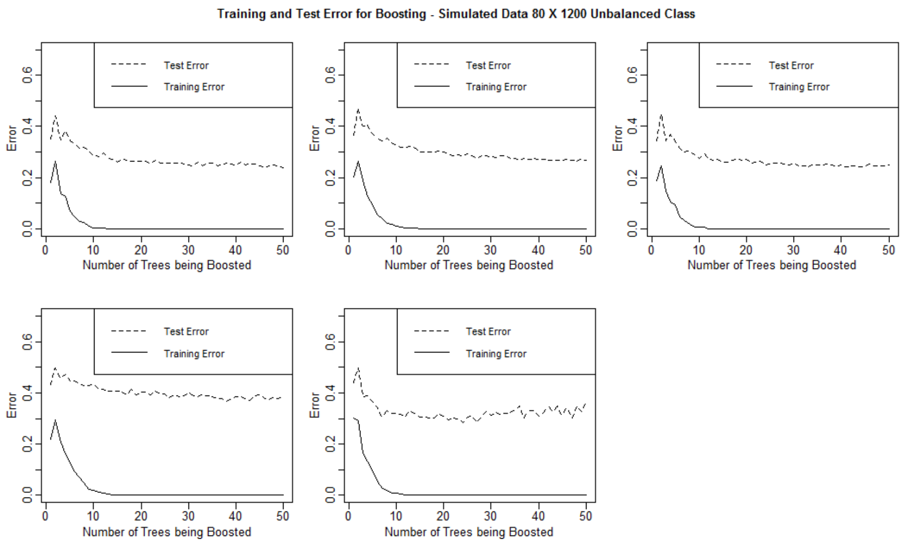

FIGURE 3.4: Training and Test Error on Imbalanced Datasets of Size 80 X 1200

3.3 Trends

The results for the *imbalanced* datasets of the sizes 250 X 17, 80 X 17, 250 X 1200, and 80 X 1200 can be found in Figures 3.1, 3.2, 3.3, and 3.4, respectively; the results of these four sizes of *balanced* datasets are in Figures 3.5, 3.6, 3.7, and 3.8, respectively.

We can see many trends in each of these figures, including the following:

- The datasets of size 250 X 17 in Figures 3.1 and 3.5 have a smaller test error than the others.
- The test error varies greatly in the datasets with a sample size of 80.
- Some of the datasets with the same attributes yield different results.

In regards to the last bullet point, reference Figure 3.2. The second graph (in the middle of the top row) shows a smaller error than the other graphs. This is very interesting because the 5 plots in this figure come from datasets with the same characteristics, so it is important to note the variability among datasets. These findings for the simulated datasets may help to explain later results; however, our conclusions will be drawn from the Multiple Linear Regression that will be done in Section 3.4.

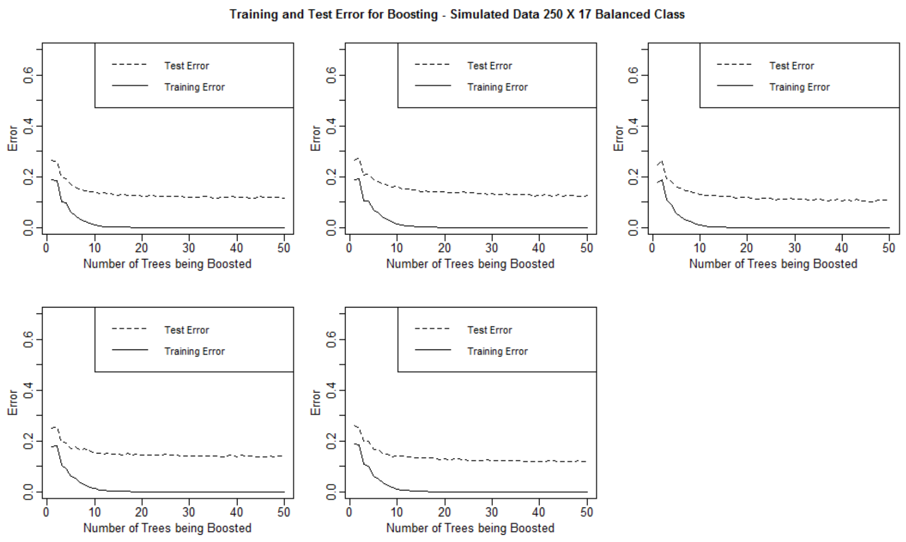

FIGURE 3.5: Training and Test Error on Balanced Datasets of Size 250 X 17

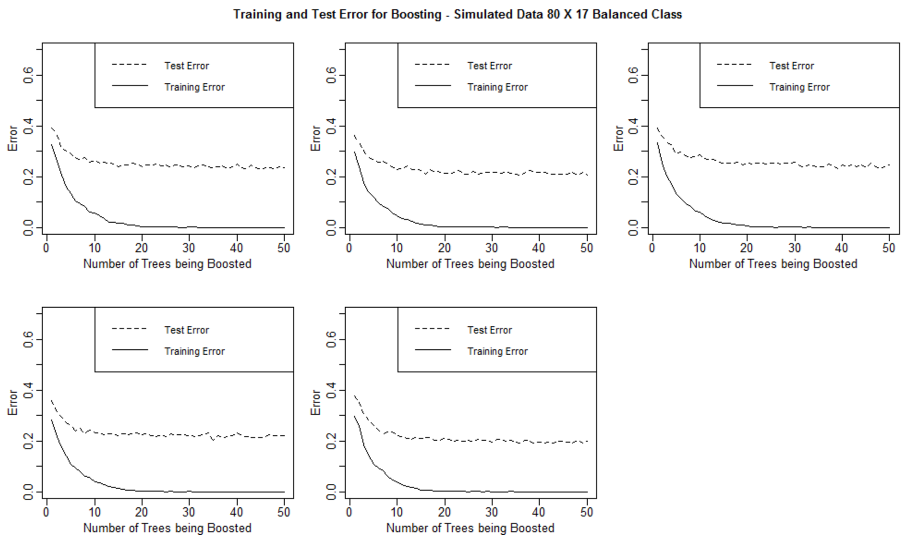

FIGURE 3.6: Training and Test Error on Balanced Datasets of Size 80 X 17

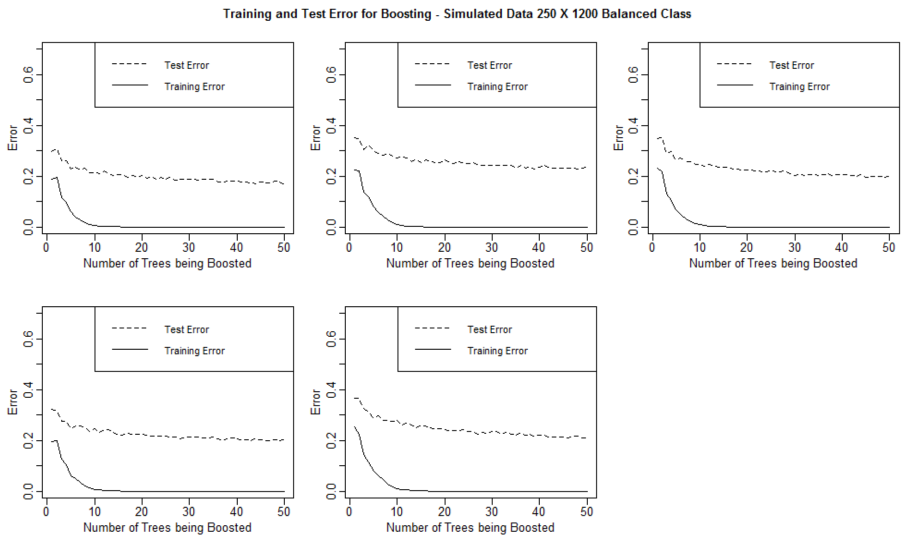

FIGURE 3.7: Training and Test Error on Balanced Datasets of Size 250 X 1200

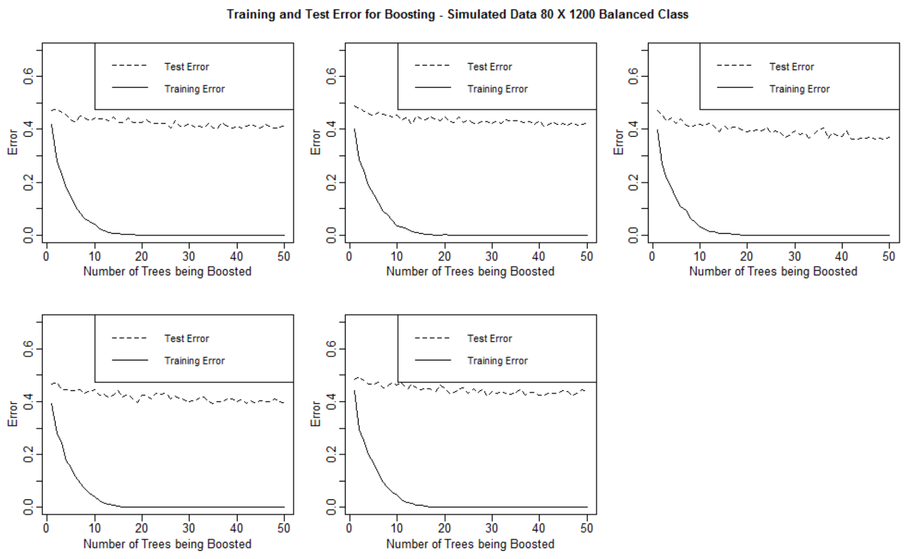

FIGURE 3.8: Training and Test Error on Balanced Datasets of Size 80 X 1200

3.4 Modeling the Test Error

3.4.1 The Variables

Now we will perform Regression Analysis on the simulated datasets. The data we are using to build this model come directly from the simulated datasets. Our response will be the average test error of each dataset after the "elbow," or after the error begins to flatten. To calculate this response value, we are averaging the mean errors over 20 replications of the boosting process explained at the end of Section 3.2. For these 20 replications, we only studied the number of trees up to $T = 30$. As mentioned in Section 2.2, it is evident that the test and training error flatten out closer to $T = 20$, so instead of requiring much more time and memory to boost the trees until $T = 50$, we stopped at $T = 30$ for these replications. Thus, the response variable we are using for this Regression Analysis is the average test error of each simulated dataset over all 20 replications, from $T = 20$ to $T = 30$. The explanatory variables also come from each of the simulated datasets; they are as follows:

- Sample Size, denoted as n
- Number of Parameters, denoted as p
- Ratio of Sample Size to Number of Parameters, denoted as np
- Number of Data Points in Class 0, denoted as z
- Number of Data Points in Class 1, denoted as o
- Ratio of Data Points in Class 0 to Data Points in Class 1, denoted as zo

Note: Since we use a stratified random sample in the training/test set split, the number of data points in Class 0 and Class 1 will be the same for each individual replication.

Our regression model will take the following form:

$$log(Y_s) = \beta_0 + \beta_1 X_{s1} + \ldots + \beta_m X_{sm} + \epsilon_s.$$

In this model,

- Y_s is the predicted average test error for simulation s;
- X_{s1} through X_{sm} are the values that the m significant explanatory variables take on from simulation s;
- β_0 through β_m are the calculated coefficients for our model;
- ϵ_s is the error, or noise, of the model for simulation s.

3.4.2 Multiple Linear Regression Models

We constructed several Multiple Linear Regression models before choosing the one that would reveal our test error. The four models are listed below:

- Full multiple linear regression (MLR) model on all 6 parameters
- Bidirectional stepwise regression, using the Bayesian Information Criterion (BIC), on the full MLR model
- Full MLR model on all 6 parameters with a logarithmic transformation on the response
- Bidirectional stepwise regression, using BIC, on the full MLR model with a logarithmic transformation on the response

The model that best fits our data, is the last one, namely the bidirectional stepwise regression on the full MLR model with a logarithmic transformation on the response. Our MLR model based on the simulated data is the following:

$$log(Y_s) = -1.268 + 0.0003683 p_s - 0.009514 z_s + \epsilon_s$$

The only significant variables are the number of parameters, p, and the number of data points classified in group 0, z. Also, we can see that the variable z has a greater affect on the test error outcome than the variable p because the coefficient is larger. These are interesting results because, as mentioned in Section 3.1, we were confident at that time that the sample size, n would be a significant variable; however, our results show otherwise. This could be due to the fact that our range of sample sizes for simulating datasets (i.e. 80 and 250) was not large enough to show an effect.

Since this model was derived with a logarithmic transformation on the response, we must exponentiate our model to return the predicted average test error instead of the log of the predicted average test error. Recall from Section 2.2 that we are modeling our test error for each simulation, Y_s to build our function ϕ. Thus, ϕ, for our case can be defined as:

$$\phi(p_s, z_s) = Y_s = \exp(-1.268 + 0.0003683p_s - 0.009514z_s + \epsilon_s).$$

To reiterate,

- Y_s is the predicted average test error for simulation s;
- p_s is the number of parameters for simulation s;
- z_s is the number of data points in class zero for simulation s;
- ϵ_s is the noise of the model for simulation s.

Recall that our intent is to use the basis of the Vapnik Chervonenkis bound to create a generalized bound for the theoretical risk of the adaptive boosting algorithm in the form of:

$$R(H) \leqslant \hat{R}_n(H) + \phi(\ldots).$$

We now have an estimated model, ϕ, said to be:

$$\phi(p, z) = Y = \exp(-1.268 + 0.0003683p - 0.009514z),$$

and we have bounded the empirical risk by the following decreasing exponential function:

$$\hat{R}_n(H) \leqslant exp\Big(-2\sum_{t=1}^{T}\gamma_t^2\Big).$$

Based on the data, we hope to conclude that the generalized error is bounded above by:

$$R(H) \leqslant exp\Big(-2\sum_{t=1}^{T}\gamma_t^2\Big) + \exp(-1.268 + 0.0003683p - 0.009514z).$$

A benefit of this generalized error bound in comparison to the bounds in Section 2.3 is that the terms in the function ϕ are more comprehensive. We can see from the bound that as the number of parameters increases, we can expect our generalized error to increase as well. In other words, this bound can give more information to one who was interested in collecting data for a study. He or she can calculate what the structure of the data should be to lower the test error bound.

In Chapter 4, we will see how this bound performs on the real datasets discussed in Section 2.2.

Chapter 4

Application to Real datasets

In this section, we will apply the generalization bound to the real datasets presented in Section 2.2. Recall that we are estimating our bound to be:

$$R(H) \leqslant exp\Big(-2\sum_{t=1}^{T}\gamma_t^2\Big) + \exp(-1.268 + 0.0003683p - 0.009514z + \epsilon),$$

and the real datasets to which we are applying the bound are the following:

- Ionosphere (Sigillito, 1989)
- Breast Cancer (Lichman, 2013b)
- Crash (Lucas et al., 2013)
- Dermatology (Lichman, 2013a)
- Letter Recognition (Slate, 1998)
- Colon Cancer (Alon et al., 1999)
- Lung Cancer (Institute, 2016)
- Lymphoma (Fokoue, 2016a)
- Lymphoma 2 (Fokoue, 2016a)
- Prostate Cancer (Fokoue, 2016b)

Figures 4.1 and 4.2 are almost identical to Figures 2.1 and 2.2. The only difference is that Figures 4.1 and 4.2 contain the generalized error bound.

Keep in mind that this bound is based on the Vapnik Chervonenkis Bound in Section 1.1, so it is probabilistic, and the projected generalization error may not always bound the test error.

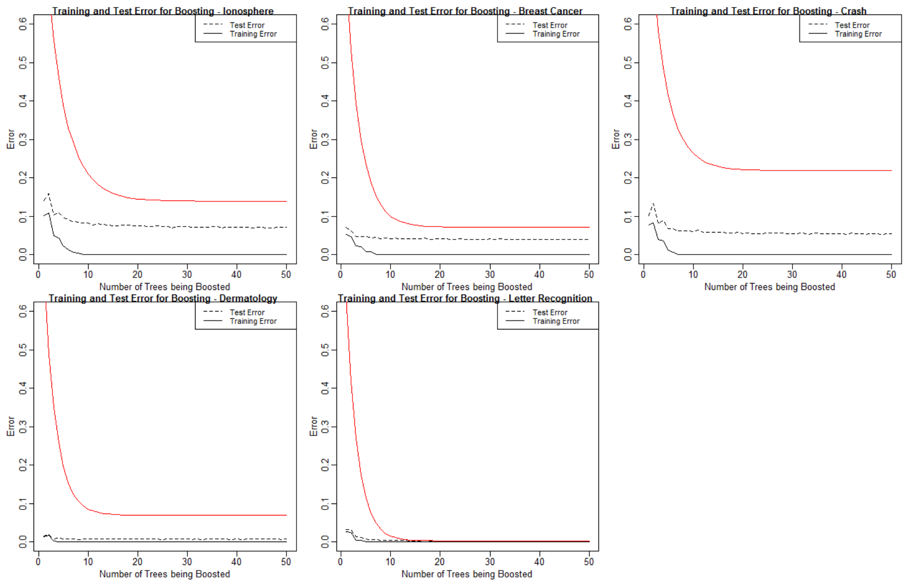

FIGURE 4.1: Large n Small p Real Datasets with Generalization Error Bound

We can see from these figures that in most cases, the generalization error bound *is* greater than our empirical test error. The one case among our examples where the generalized error fails to bound the test error is in the Prostate Cancer dataset. These results are shown in the bottom middle graph of Figure 4.2.

Although it is hard to tell, the theoretical error in the Letter Recognition dataset, located in the bottom middle graph of Figure 4.1, is bounding the test error. The calculated theoretical error plateaus at 0.00321, and the empirical error plateaus around 0.00244. For the other eight datasets, it is easy to tell that the generalized error *is* bounding our empirical results.

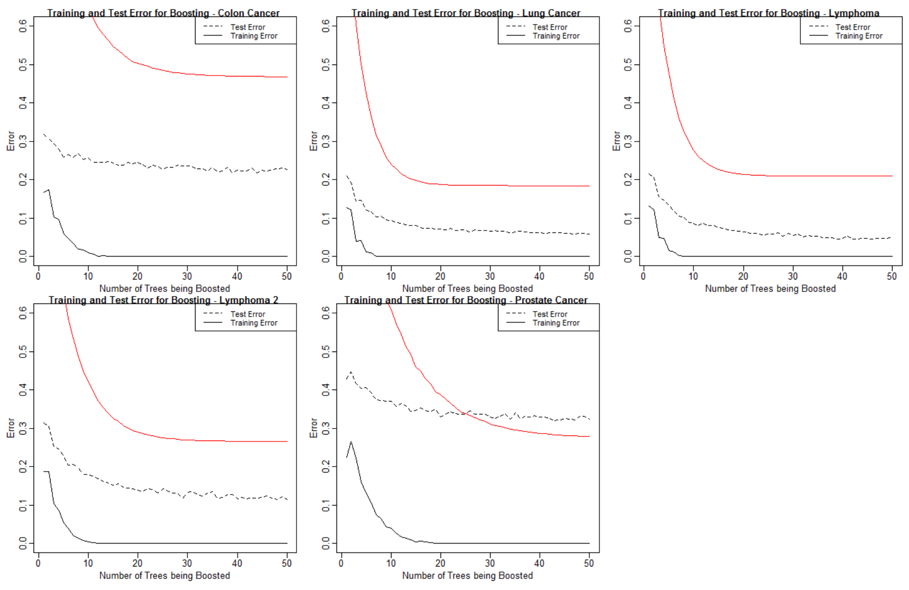

FIGURE 4.2: Large p Small n Real Datasets with Generalization Error Bound

Chapter 5

Conclusion

5.1 Restating The Problem

The main focus of this thesis is to model the generalized error. As previously stated in Section 1.3, our goal is to be able to suggest a bound for the generalized error of the Boosting Algorithm. We first boosted ten real datasets of different sizes and characteristics with a binary response, and plotted the error from each dataset.

By studying and analyzing the empirical demonstrations of the ten datasets, we gathered some conjectures about which characteristics of the datasets seemed to be causing the changes in test error results. We concluded that it appears that the following characteristics are important:

- Sample size
- Number of parameters
- Ratio of sample size to number of parameters
- Number of data points in Class 0
- Number of data points in Class 1
- The ratio of data points in Class 0 to data points in Class 1

We strategically simulated datasets by varying the above-listed characteristics, so we can collect more information on exactly what alters the test error and by how much.

After that, we boosted those 40 datasets with 20 replications and used that data to build a regression model for the test error. Our bidirectional stepwise multiple linear regression of the 6 parameters with a logarithmic transformation of the response pointed to two significant variables. The significant variables are the number of parameters in the dataset and the number of observations in the dataset that had an observed classification as Class 0.

This model, explicitly stated, is:

$$\phi(p, z) = Y = \exp(-1.268 + 0.0003683p - 0.009514z + \epsilon),$$

where:

- Y is the predicted average test error;
- p is the number of parameters;
- z is the number of data points in class zero;
- ϵ is the noise of the model.

After following the basis of the Vapnik Chervonenkis bound in Theorem 1, we derived our proposed generalized error bound of the Boosting Algorithm. It is as follows:

$$R(H) \leqslant exp\Big(-2\sum_{t=1}^{T}\gamma_t^2\Big) + \exp(-1.268 + 0.0003683p - 0.009514z + \epsilon).$$

5.2 Impact of the Generalized Error Bound

As explained in Section 2.3, the existing generalized error bounds are quite technical. The variables used in these bounds are in-depth and can be hard to understand. For example, the VC dimension defined by ζ can get very complex depending on the function space defined. In the case of the generalized error bound derived in this thesis, the variables (i.e., the number of parameters in the dataset that is being boosted, and number of observations classified in Class 0) are easy to comprehend. This is beneficial because if one were conducting a study, he or she will know to limit the number of parameters used in the research because our bound shows that increasing the number of parameters is going to increase the upper bound of the boosting algorithm's test error.

5.3 Future Results

If one were inclined to continue research to make this test error bound more accurate, we would suggest simulating more datasets of different sizes. We used 4 sizes of data sets: 250 X 17, 80 X 17, 250 X 1200, and 80 X 1200. We simulated 5 datasets of each size with an imbalanced binary response and 5 datasets of each size with a balanced binary response. Had we had more resources, we could have simulated datasets with a larger sample size. The largest sample size of the real datasets that were studied is $n = 1573$. If a larger range of sample sizes had been covered, we may have been able to get even more generalizable results without extrapolation.

Although the sample size was not a significant variable in the final model of the test error, perhaps analyses involving a larger range of sample sizes might reveal that sample size is indeed significant. Still, a moderate range of sample sizes were covered and results of the shift portrayed in Figures 4.1 and 4.2 in Chapter 4 show that our probabilistic bound still captures our test error in 9 out of the 10 real examples. Since the generalization error is a probabilistic bound, we are confident about the results.

Appendix A

R Functions

```
winds <- function(nrow, ncol, maxsize=6, aryx=1,
                cex.lab=1.2, cex.axis=1.2, title=FALSE,
        mars=c(3,3.2,1,1)+.1, omas=c(0,0,0,0),
        mgps=c(2,0.7,0), mar3titlePlus=2,
        byrow=TRUE)
{
  graph.ar <- aryx*nrow/ncol
  if(graph.ar > 1) {rsize <- maxsize;
                csize <- maxsize/graph.ar} else
                   {csize <- maxsize;
                rsize <- maxsize*graph.ar}
  windows(csize,rsize)
  if(byrow) par(mfrow=c(nrow,ncol)) else
            par(mfcol=c(nrow,ncol))
  par(mar=c(mars[1], mars[2], ifelse(title,
                mars[3]+mar3titlePlus,
                mars[3]), mars[4]),
        mgp=mgps, oma=if(length(omas)==1)
            c(0,0,omas,0) else if(length(omas)==2)
            c(omas[1], 0, omas[2], 0) else omas,
        cex.lab=cex.lab, cex.axis=cex.axis)
  invisible()
}

boosted.trees <- function(xy, nbase)
{
```

```
Y        <- xy$Y
n        <- nrow(xy)
m        <- round(0.75*n)
p        <- ncol(xy)-1
T        <- nbase

alpha    <- numeric(T)
epsilon  <- numeric(T)
weight   <- rep(1/n, n)
h        <- NULL

for(t in 1:T)
{
   decent.base.learner <- 0
   while(!decent.base.learner)
   {
      boost.id     <- sample(1:n, m,
                        replace=T, prob=weight)
      base.learner <- rpart(as.factor(Y)~.,
                        data=xy, subset=boost.id)
      yhat         <- predict(base.learner,
                        xy[,-(p+1)], type='class')
      epsilon.candidate <- error.weighted(Y,
              yhat, weight)
      ifelse(epsilon.candidate==0,
              epsilon.candidate <- 0.0001,
              epsilon.candidate <- epsilon.candidate)
      decent.base.learner <- (epsilon.candidate < 0.5)
   }
   epsilon[t] <- epsilon.candidate
   alpha[t]   <- (1/2)*log((1-epsilon[t])/epsilon[t])
   weight     <- weight*exp(alpha[t]*indicator(Y!=yhat))

weight   <- weight/sum(weight)
h        <- c(h, list(base.learner))
```

```
  }
  return(list(alpha=alpha, h=h, weight=weight))
}

predict.boosted.tree <- function(h, alpha, xnew)
{
  T       <- length(h)
  nnew    <- nrow(xnew)
  h.t.x   <- NULL
  for(t in 1:T){h.t.x <- cbind(h.t.x,
                  pm(predict(h[[t]],xnew,
          type='class')))}
  m.alpha <- matrix(rep(alpha, nnew),
                  byrow=T, nrow=nnew)
  return(ifelse(rowSums(m.alpha*h.t.x)<0.5,0,1))
}

error.weighted <- function(y, yhat, weight)
{
  err <- ifelse(y!=yhat, 1,0)
  return(sum(err*weight))
}

indicator <- function(predicate)
{
  return(ifelse(predicate, 1,0))
}

pm <- function(y)
{
  uy <- sort(unique(y))
  return(ifelse(y==uy[1],-1,+1))
}

measures <- function(label, response)
```

```
{
  n               <- length(label)
  confmat       <- table(label, response)
  Accuracy      <- sum(diag(confmat))/n
  FPR           <- confmat[1,2]/rowSums(confmat)[1]
  TPR           <- confmat[2,2]/rowSums(confmat)[2]
  FNR           <- confmat[2,1]/rowSums(confmat)[2]
  TNR           <- confmat[1,1]/rowSums(confmat)[1]
  Precision     <- confmat[2,2]/colSums(confmat)[2]
  Recall        <- TPR
  Specificity <- TNR
  Sensitivity <- TPR
  F.measure     <- 2*(Precision*Recall)/(Precision+Recall)
  measured      <- list(Accuracy=Accuracy,
                       Precision = Precision,
                       F.measure = F.measure,
                       Recall=Recall,
                       FPR=FPR, TPR = TPR,
                       FNR=FNR, TNR = TNR,
                       Specificity=Specificity,
                       Sensitivity=Sensitivity)
  return(measured)
}

extract<-function(measure)
{
  v      <- numeric(10)
  v[1]   <- measure$Accuracy
  v[2]   <- measure$Precision
  v[3]   <- measure$Recall
  v[4]   <- measure$F.measure
  v[5]   <- measure$FPR
  v[6]   <- measure$TPR
  v[7]   <- measure$FNR
  v[8]   <- measure$TNR
  v[9]  <- measure$Specificity
  v[10] <- measure$Sensitivity
  return(v)
}

simclass <- function(n=300,p=10,seed=NULL)
{
  set.seed(seed)
  mu      <- round(runif(p, 0,1),2)
  sigma   <- round(runif(p,1,3),2)
  sigma.m <- `diag <-` (matrix(0, p, p), 1)*sigma
  X       <- mvrnorm(n,mu,sigma.m)
  y       <- X[,1]+X[,4]+X[,5]
  pii     <- 1/(1+exp(-y))
  y.c     <- rbinom(n,1,pii)
  table(y.c)
  sim.dat.c              <- data.frame(cbind(y.c,X))
  names(sim.dat.c)[1] <- "Y"
  return(sim.dat.c)
}

simclasseven <- function(n=300,p=10,seed=NULL)
{
  set.seed(seed)
  mu      <- round(runif(p, 0,1),2)
  sigma   <- round(runif(p,1,3),2)
  sigma.m <- `diag <-` (matrix(0, p, p), 1)*sigma
  X       <- mvrnorm(n,mu,sigma.m)
  y       <- X[,1]+X[,4]+X[,5]
  pii     <- 1/(1+exp(-y))
  y.c     <- ifelse(pii<=median(pii),0,1)
  table(y.c)
  sim.dat.c              <- data.frame(cbind(y.c,X))
  names(sim.dat.c)[1] <- "Y"
  return(sim.dat.c)
```

```
}
```

(Voelkel, 2014, Fokoue, 2015b, Ramos, 2016).

Bibliography

Abney, Steven, Robert Schapire, and Yoram Singer (1999). *Boosting Applied to Tagging and PP Attachment*. Tech. rep. Florham Park, NJ: AT&T Labs.

Alfaro, Esteban, Matias Gamez, and Noelia Garcia (2013). “adabag: An R Package for Classification with Boosting and Bagging”. In: *Journal of Statistical Software* 54.2.

Alfaro, Esteban et al. (2015). *Package ‘adabag’*. URL: `https://cran.r-project.org/web/packages/adabag/adabag.pdf`.

Alfaro-Cortes, Esteban et al. (2016). *R Documentation - Applies Multiclass AdaBoost.M1, SAMME and Bagging*. URL: `http://127.0.0.1:28781/library/adabag/html/adabag-package.html`.

Alon, U. et al. (1999). “Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays”. In: *Proceedings of the National Academy of Sciences of the United States of America* 96, pp. 6745–6750.

Appel, Ron et al. (2013). *Quickly Boosting Decision Trees - Pruning Underachieving Features Early*. Tech. rep. 28. JMLR Workshop and Conference Proceedings.

Drucker, Harris and Corinna Cortes (1996). “Boosting Decision Trees”. In: *Advances in Neural Information Processing Systems* 8.

Drucker, Harris, Robert Schapire, and Partice Simard (1993). “Improving Performance in Neural Networks Using a Boosting Algorithm”. In: *Advances in Neural Information Processing Systems* 5, pp. 42–49.

Fokoue, Ernest (2015a). *Lecture Notes for Principles of Statistical Data Mining*.

— (2015b). *R Code for various Functions*. Rochester Institute of Technology Principles of Data Mining Course.

— (2016a). *Lymphoma Dataset*. Lymphoma Dataset.

— (2016b). *Prostate Cancer Dataset*. Prostate Cancer Dataset.

Freund, Yoav and Robert E. Schapire (1996). *Experiments with a New Boosting Algorithm*. Paper presented at Machine Learning: Proceedings of the Thirteenth International Conference. Murray Hill: AT&T Laboratories.

Institute, Broad (2016). *Cancer Program Legacy Publication Resources - Lung Cancer Data*. URL: `http://www.broadinstitute.org/cgi-bin/cancer/publications/view/87`.

James, Gareth et al. (2013). *An Introduction to Statistical Learning with Applications in R*. New York: Springer Science+Business Media.

Koltchinskii, Vladimir, Dmitriy Panchenko, and Fernando Lozano (2001). "Some New Bounds on the Generalization Error of Combined Classifiers". MA thesis. University of New Mexico.

Kuhn, Max (2015). *A Short Introduction to the Caret Package*. URL: `https://cran.r-project.org/web/packages/caret/vignettes/caret.pdf`.

Lichman, M. (2013a). *UCI Machine Learning Repository - Dermatology Data Set*. URL: `http://archive.ics.uci.edu/ml/datasets/Dermatology`.

— (2013b). *UCI Machine Learning Repository – Breast Cancer Data Set*. URL: `http://archive.ics.uci.edu/ml/datasets/Dermatology`.

Loh, Wei yin (2011). "Classification and Regression Trees". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, pp. 14–23.

Longadge, Rushi, Snehlata S. Dongre, and Latesh Malik (2013). "Class Imbalance Problem in Data Mining: Review". In: *International Journal of Computer Science ad Network* 2.1.

Lucas, D. et al. (2013). "Failure Analysis of Parameter-Induced Simulation Crashes in Climate Models". In: *Geoscientific Model Development*, pp. 585–623.

Margineantu, Dragos D. and Thomas G. Dietterich (1997). *Pruning Adaptive Boosting*. Tech. rep. Oregon State University.

Mazurowskia, Maciej A. et al. (2008). "Training Neural Network Classifiers for Medical Decision Making: The Effects of Imbalanced Datasets on Classification Performance". In: *Neural Networks* 21.2-3, pp. 427–436.

Melnykov, Volodymyr, Wei-Chen Chen, and Ranjan Maitra (2012a). "MixSim: An R Package for Simulating Data To Study Performance of Clustering Algorithms". In: *Journal of Statistical Software* 51.12.

Melnykov, Volodymyr, Wei-Chen Chen, and Ranjan Maitra (2012b). *simdataset*. URL: `http://finzi.psych.upenn.edu/library/MixSim/html/simdataset.html`.

Melnykov, Volodymyr et al. (2015). *MixSim: Simulating Data to Study Performance of Clustering Algorithms*. URL: `https://cran.r-project.org/web/packages/MixSim/index.html`.

Observational Medical Outcomes Partnerships (2013). URL: `http://omop.org/node/70`.

Ohio, State University (2015). *Boosting*. URL: `http://www.stat.osu.edu/~yklee/boosting.pdf`.

Quinlan, J. Ross (2006). *Bagging, Boosting, and C4.5*. Paper presented at Proceedings, Forteenth National Conference on Artficial Intellegence. Sydney: University of Sydney.

R, Team Core (1993a). *Median Value*. URL: `https://stat.ethz.ch/R-manual/R-devel/library/stats/html/median.html`.

— (1993b). *The Binomial Distribution*. URL: `https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Binomial.html`.

— (2016). *R: A Language and Environment for Statistical Computing*. http://www.R-project.org/. R Foundation for Statistical Computing.

Ramos, Andre Lobato (2016). *Simulating Data Code*.

Revelle, William (2006). *Generate simulated data structures for circumplex, spherical, or simple structure*. URL: `http://www.personality-project.org/r/html/sim.item.html`.

— (2015). *Procedures for Psychological, Psychometric, and Personality Research*. URL: `https://cran.r-project.org/web/packages/psych/psych.pdf`.

Sarkar, Deepayan (2015). *Trellis Graphics for R*. URL: `https://cran.r-project.org/web/packages/lattice/lattice.pdf`.

Schapire, Rob and Alina Ene (2006). *Foundations of Machine Learning Lecture 10*.

Schapire, Robert and Yoram Singer (1999). "Improved Boosting Algorithms Using Confidence-rated Predictions". In: *Machine Learning* 37, pp. 297–336.

— (2000). "BoosTexter: A Boosting-based System for Text Categorization". In: *Machine Learning* 39, pp. 135–168.

Schapire, Robert E. (1999). "A Brief Introduction to Boosting". In: *Conference on Artificial Intellegence*. AT&T Labs, pp. 1–6.

Schapire, Robert E. and Yoav Freund (2012). *Adaptive Computation and Machine Learning : Boosting : Foundations and Algorithms*. Cambridge: MIT Press.

Sigillito, Vince (1989). *R Documentation Johns Hopkins University Ionosphere database*. URL: `http://127.0.0.1:28781/library/mlbench/html/Ionosphere.html`.

Slate, David (1998). *R Documentation Letter Image Recognition Data*. URL: `http://127.0.0.1:15163/library/mlbench/html/LetterRecognition.html`.

Therneau, Terry, Beth Atkinson, and Brian Ripley (2015). *Recursive Partitioning and Regression Trees*. URL: `https://cran.r-project.org/web/packages/rpart/rpart.pdf`.

Touloumis, Anestis (2015). *Shrinkage Covariance Matrix Estimators*. URL: `https://cran.r-project.org/web/packages/ShrinkCovMat/ShrinkCovMat.pdf`.

Vaughan, Jennifer Wortman et al. (2006). *Machine Learning Theory Lecture 14: Generalization Error of Adaboost*.

Voelkel, Joseph G. (2014). *Winds Code for Plotting*. Rochester Institute of Technology Statistical Software Course.