UNIVERSITY OF CINCINNATI

Date: May 03, 2007

I, Mark Crisanti, hereby submit this work as part of the requirements for the degree of:

Master of Science
in:
Department of Environmental Health

It is entitled:
The Predictive Accuracy of Boosted Classification Trees Relative to Discriminant Analysis and Logistic Regression

This work and its defense approved by:

Chair: Paul A. Succop
Linda S. Levin
Marepalli B. Rao
The Predictive Accuracy of Boosted Classification Trees Relative to Discriminant Analysis and Logistic Regression

A thesis submitted to

Department of Environmental Health
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of

Master of Science in Biostatistics

in the College of Medicine

2007

by

Mark Crisanti

Bachelor of Science in Chemistry, Villanova University, 1975
Doctor of Philosophy in Chemistry, The Pennsylvania State University, 1980

Committee Chair: Paul A. Succop, Ph.D.
Abstract

This research used simulated data to compare the predictive accuracy of three binary classification methods: 1) boosted classification trees; 2) discriminant analysis; and 3) logistic regression (LR). The simulations involved 10 predictor variables for both cases and controls. Prediction error rates for test sets were measured using three types of decision boundaries: 1) a linear boundary from normal populations; 2) a quadratic boundary from normal populations; and 3) a quadratic boundary from non-normal populations. Results showed that discriminant analysis yielded predictive accuracies that were higher than boosted classification trees for all three decision boundaries. The performance of LR relative to boosted trees depended upon the degree of misspecification of the LR model and upon the size of the training data sets. The use of boosted classification trees may be favored in situations where training data are plentiful and the underlying physical model is poorly understood.
Acknowledgment

I wish to thank my advisor, Dr. Paul Succop, for his guidance throughout my graduate studies. Not only did he patiently answer questions about my coursework and projects, he also created several consulting opportunities to help me learn how to apply my knowledge in practical settings.
# Table of Contents

Abstract iii

Acknowledgment iv

Table of Contents vi

List of Tables and Figures viii

List of Symbols and Abbreviations x

Introduction 1

Overview 1

Specific Aims of This Research 5

Methods 9

The Three Classification Methods 9

Classification Trees 9

Logistic Regression 12

Discriminant Analysis 13

Boosting Algorithm 14

The Three Classification Scenarios 17

Scenario 1: Linear Boundary, Normal Populations 18

Scenario 2: Quadratic Boundary, Normal Populations 19

Scenario 3: Quadratic Boundary, Non-Normal Populations 22

Conducting the Simulation Studies 23

Statistical Analyses and Power Calculations 26
# Table of Contents (continued)

Results

<table>
<thead>
<tr>
<th>Scenario 1: Linear Boundary, Normal Populations</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large Training Sets (n = 2,000)</td>
<td>28</td>
</tr>
<tr>
<td>Medium-Sized Training Sets (n = 200)</td>
<td>34</td>
</tr>
</tbody>
</table>

Scenario 2: Quadratic Boundary, Normal Populations

| Large Training Sets (n = 2,000)               | 40 |
| Medium-Sized Training Sets (n = 200)         | 47 |

Scenario 3: Quadratic Boundary, Non-Normal Populations

| Large Training Sets (n = 2,000)               | 52 |
| Medium-Sized Training Sets (n = 200)         | 57 |

Discussion

| Choice of Base Classifiers                   | 60 |
| Comparison of Boosted and Stable Methods    | 62 |
| Study Limitations                           | 64 |
| Future Research                             | 66 |

Conclusions

|                                     | 68 |

Bibliography

|                                   | 70 |

Appendix 1

|                                            | 77 |

Appendix 2

|                                            | 83 |

Appendix 3

|                                            | 84 |
List of Tables and Figures

Tables

Table 1. Comparison of Minimum Test Error Rates by Classification Method: Scenario 1 with Large Training Sets 31
Table 2. Comparison of Minimum Test Error Rates by Classification Method: Scenario 1 with Medium-Sized Training Sets 39
Table 3. Comparison of Minimum Test Error Rates by Classification Method: Scenario 2 with Large Training Sets 42
Table 4. Comparison of Minimum Test Error Rates by Classification Method: Scenario 2 with Medium-Sized Training Sets 48
Table 5. Comparison of Minimum Test Error Rates by Classification Method: Scenario 3 54

Figures

Figure 1. Examples of One-Level (Stumps) and Two-Level Classification Trees 11
Figure 2. Summary of the Real AdaBoost Algorithm 16
Figure 3. Hypothesized Ordering of Test Error Rates by Classification Method 23
Figure 4. Summary of the Steps Performed in the Data Simulations 25
Figure 5. Scenario 1: \( \mu_1 = (0.25, 0.25, \ldots, 0.25) \) for Large Training Sets 29
Figure 6. Scenario 1: \( \mu_1 = (0.50, 0.50, \ldots, 0.50) \) for Large Training Sets 32
Figure 7. Scenario 1: \( \mu_1 = (0.75, 0.75, \ldots, 0.75) \) for Large Training Sets 33
Figure 8. Scenario 1: \( \mu_1 = (0.25, 0.25, \ldots, 0.25) \) for Medium-Sized Training Sets 35
List of Tables and Figures (continued)

Figure 9. Scenario 1: \( \mu_1 = (0.50, 0.50, \ldots, 0.50) \) for Medium-Sized Training Sets 36

Figure 10. Scenario 1: \( \mu_1 = (0.75, 0.75, \ldots, 0.75) \) for Medium-Sized Training Sets 37

Figure 11. Scenario 2: \( \mu_1 = (0.1, 0.1, \ldots, 0.1) \) for Large Training Sets 41

Figure 12. Scenario 2: \( \mu_1 = (0.5, 0.5, \ldots, 0.5) \) for Large Training Sets 45

Figure 13. Scenario 2: \( \mu_1 = (0.9, 0.9, \ldots, 0.9) \) for Large Training Sets 46

Figure 14. Scenario 2: \( \mu_1 = (0.1, 0.1, \ldots, 0.1) \) for Medium-Sized Training Sets 49

Figure 15. Scenario 2: \( \mu_1 = (0.5, 0.5, \ldots, 0.5) \) for Medium-Sized Training Sets 50

Figure 16. Scenario 2: \( \mu_1 = (0.9, 0.9, \ldots, 0.9) \) for Medium-Sized Training Sets 51

Figure 17. Scenario 3 for Large Training Sets 53

Figure 18. Dependence of the Mean Test Error Rate upon the Number of Properly Specified Terms in the LR Model for Scenario 3 with Large Training Sets 56

Figure 19. Scenario 3 for Medium-Sized Training Sets 58

Figure 20. Dependence of the Mean Test Error Rate upon the Number of Properly Specified Terms in the LR Model for Scenario 3 with Medium-Sized Training Sets 59
List of Symbols and Abbreviations

Symbols

$N_{10}(\mu, \Sigma)$ - 10-dimensional multivariate normal distribution having a vector of means, $\mu$, and a variance-covariance matrix, $\Sigma$.

$\mu_0$ - vector of population means for controls

$\mu_1$ - vector of population means for cases

$I$ - identity matrix

$\Sigma_0$ - variance-covariance matrix for the population of controls

$\Sigma_1$ - variance-covariance matrix for the population of cases

$\sim$ - distributed as (e.g., $Y \sim N_{10}(\mu, \Sigma)$)

$\chi^2(n)$ - chi-square distribution with $n$ degrees of freedom

$\Phi$ - cumulative normal distribution function

$\alpha$ - alpha, probability of type I error

$1 - \beta$ - power of a statistical test

Abbreviations

LDA - linear discriminant analysis

LR - logistic regression

QDA - quadratic discriminant analysis
Introduction

Overview

Binary classification is an important problem in both diagnostic medicine and environmental health. Its applications include the screening and diagnosis of sub-clinical disease, modeling risk factors for future disease, and identifying individuals who may be responsive to a therapeutic treatment, among others. Classification problems are growing in importance because recent advances in genomics have led to the identification of potential biomarkers for numerous diseases, as well as to the identification of genes that may increase an individual’s susceptibility to various environmental risk factors. Thus, increasing the predictive power of classification methods is an important endeavor.

A binary classification rule, also known as a classifier, is developed by presenting a set of training data to a suitable classification method. The training data consist of a set of \( n \) observations, \((y_1, x_1), (y_2, x_2), \ldots, (y_n, x_n)\), where \( x_i \) represents the vector of potential predictor variables for the \( i \)th observation and \( y_i \) represents the known binary class label for that observation. Several methods exist for developing classification rules from the training data. These methods include logistic regression, classification trees (i.e., binary recursive partitioning), discriminant analysis, artificial neural networks, support vector machines, and nearest-neighbor approaches. A systematic comparison of these classification methods has recently been made by Hastie, Tibshirani, and Friedman.

Among the more important advances in classification methodology has been the development of a procedure called boosting. While most classification methods use a single classifier to assign observations to classes, boosting uses a linear combination of classifiers to make its assignments. Each of the classifiers in the combination is trained on a different version
of the training set. In the first step of the boosting procedure, a base classifier of choice (e.g., a classification tree) is used to develop a classification rule in the usual way, assigning equal weights to each observation in the training data set. In each subsequent step of the boosting procedure, the training data are reweighted to give more weight to those training observations that were misclassified in the previous step. This reweighting of the data forces the base classifier to focus more intently on the harder-to-classify observations at each progressive step of the boosting procedure. This process of reweighting and reclassifying the training data continues for a preset number of iterations, creating a linear combination of classifiers. The final class assignment for an observation is made by a weighted majority vote of the ensemble of classifiers. A variety of boosting algorithms have been proposed for reweighting the training data and for choosing the linear coefficients of the classifiers.26

Because boosting uses an ensemble of classifiers to make its class assignments, it is often better than a single classifier at fitting complex decision boundaries. Several empirical studies have demonstrated the extent to which boosting can improve a classifier’s predictive accuracy.27-29 For example, Breiman has shown that for a variety of empirical and simulated data sets, boosting reduced misclassification rates by 25% to 75% relative to single classifiers of the same type.30

While boosting is somewhat flexible with respect to the choice of base classifier, not all classification methods benefit from boosting. Breiman has proposed that the base classifiers used in boosting must be “unstable.”31 A classification method is considered unstable if a small change in the training data set can induce a large change in the derived classifier.32 When the base classifiers are unstable, the reweighting of the data after each boosting step produces a set of classifiers that is diverse (i.e., each classifier emphasizes different subsets of the predictor
variables). This diversity gives the ensemble of classifiers increased flexibility to fit a complex decision boundary. Examples of unstable classification methods include classification trees, neural nets, and multivariate splines.

Breiman has suggested that boosting is unproductive with stable classification methods, such as discriminant analysis, logistic regression, and nearest-neighbor approaches.\textsuperscript{30-31} The classifiers developed by stable classification methods are not much affected by resampling of the training data set. For example, logistic regression is a popular classification method because of its stability; its parameter estimates are relatively constant regardless of the relative proportion of cases and controls present in the training data.\textsuperscript{33} Because the boosting of stable methods tends to produce a set of classifiers that are similar to one another, the set of classifiers has a predictive accuracy that is similar to a single classifier. Breiman and others have directly demonstrated that boosting fails to improve the predictive accuracy of linear discriminant analysis (LDA) and nearest-neighbor approaches.\textsuperscript{30-31,34}

An ensemble of unstable classifiers is at best, however, a piece-wise approximation to a decision boundary between classes. A key question then is how does the predictive accuracy of this approximate method compare to the accuracy of stable classification methods, which often exhibit high predictive accuracy when properly applied? Or more to the point, what conditions would favor the use of boosting over a stable classification method? Relatively few studies have directly addressed the topic of comparing the predictive accuracy of boosted methods to stable classification methods. That topic will be the main subject of this thesis. A brief overview of the literature provides some insights into the relative accuracy of boosted classifiers and stable methods.
One of the most widely used stable classification methods is linear discriminant analysis (LDA). Breiman showed that boosted classification trees outperformed LDA for three out of the four empirical data sets that he tested. His explanation for the poorer performance of LDA was as follows:

“The problem with LDA is usually bias; when it is wrong, it is consistently wrong, and with a simple model there is no hope of low bias across a variety of complex data sets.”

In other words, LDA can model only a simple decision boundary whereas most data sets require the modeling of a complex decision boundary. Breiman’s work suggests that boosting of unstable classifiers would be preferred in situations when a complex decision boundary exists between classes. However, Breiman offered no insight into how predictive accuracy might be improved by the use of quadratic discriminant analysis (QDA), another stable classification method that is capable of modeling more complex decision boundaries than LDA.

Dudoit et al. found that boosted decision trees had increased predictive accuracy compared to LDA for the classification of tumors for three gene expression data sets. However, when they applied QDA to these same data, they found QDA provided a predictive accuracy that was similar to boosted trees. Similarly, Wu et al. applied both LDA and QDA to mass spectrometry data for the classification of ovarian cancer data. For their application, LDA had an increased predictive accuracy compared to QDA. In addition, they found that LDA and boosted classification trees had similar predictive accuracy. The above studies suggest that, in many instances, discriminant analysis and boosted trees offer similar predictive power. However, boosted trees may have an edge when more complex decision boundaries exist between classes.
Two sets of investigators have found that the nearest-neighbor approach, another stable method, offered predictive accuracy similar to boosted trees for the classification of cancer using multispectral imaging data and gene expression data.\textsuperscript{37-38} The picture is somewhat less consistent when one compares logistic regression (LR) to boosting. Li et al. found that LR had predictive accuracy superior to boosted classification trees for selecting pre-screening items for dementia.\textsuperscript{39} In this situation, however, the poorer performance of boosting may be due to overfitting caused by the use of an overly complex base classifier. This explanation was supported by the observation that their boosted trees had a lower predictive accuracy than a single classification tree. In contrast, Neumann et al. found that boosting offered improved predictive power relative to logistic regression for two applications—predicting mortality in intensive care units and identifying potentially avoidable hospital readmissions.\textsuperscript{40} Neumann et al. offered no explanation as to why boosted trees gave better performance in this situation.

In summary, it appears that the predictive accuracy of boosted classifiers and stable classification methods are similar in many cases. Differences between the methods seem to arise when a classification method is misapplied, such as when an unnecessarily complex base classifier is used in boosting. A complex decision boundary between classes may favor boosted classifiers over some stable classification methods.

**Specific Aims of This Research**

The purpose of this research is to clarify the conditions which would favor the use of boosted classifiers over stable classification methods, and vice versa. Factors that could influence a method’s predictive accuracy include the complexity of the decision boundary, the nature of the classification method, the size of the training data set, the degree of distributional overlap
between cases and controls, use of the proper method variant (e.g., QDA v. LDA), proper model specification (as in LR), the signal-to-noise ratio, and the dimensionality of the classification problem. For boosting, additional factors could include the number of boosting iterations, the nature and complexity of the base classifiers, and the choice of method used to prevent overfitting the data. By using data simulations, I plan to illustrate the role of some of these factors in determining the relative predictive accuracy of boosted and stable classification methods.

This research will compare the predictive accuracy of boosted classification trees with two stable classification methods—logistic regression and discriminant analysis. Trees were chosen as the base classifiers for boosting because they offer a number of practical advantages. Classification trees can use both numeric and categorical variables as predictors, are robust to data outliers, can handle irrelevant data inputs, and can be constructed quickly from a computational viewpoint.\textsuperscript{22(p. 313)} Because of these advantages, classification trees are the most commonly used base classifiers in boosting.

However, there is also a theoretical reason for selecting classification trees as base classifiers for this research. Application of the Neyman-Pearson lemma to diagnostic methods has shown that classification methods that model the risk score function, \( RS(Y) = P(D=1|Y) \), provide an optimal combination of predictor variables for classification.\textsuperscript{41} This result arises because the risk score is a monotone function of the likelihood ratio function. Trees and boosted trees, along with logistic regression, neural networks, and support vector machines model the risk score in developing classifiers.\textsuperscript{42} Thus, classification trees are used in this research not only for their practical advantages but also because they may yield misclassification rates that approach the minimum misclassification rates possible.
Logistic regression was chosen as one of the stable classification methods due to both its popularity and its ability to model the risk score when developing classifiers. However, logistic regression will provide an optimal combination of predictors only if the regression model selected is a monotone function of the true underlying model. So, logistic regression offers the opportunity to see how far its predictive accuracy departs from optimality when the model is intentionally misspecified. Model misspecifications can occur unintentionally when developing models for real data sets.

Discriminant analysis was chosen as the second stable classification method also because of its popularity as a classification method and because it does not model the risk score in creating its classifiers. Instead, discriminant analysis maximizes the ratio of the between-class variance to the within-class variances. Only in the particular case that both the populations of cases and controls are multivariate normal will discriminant analysis, coincidentally, model the risk score and provide optimal combinations of predictors. Thus, discriminant analysis provides the opportunity to observe departures from optimality when cases and controls arise from non-normally distributed populations.

The accuracy of the three classification methods will be compared under three different simulation scenarios that vary in the complexity of their decision boundary and in the choice of the underlying distributions for the cases and the controls. The three scenarios include:

1. A linear boundary with independent, normally distributed predictor variables
2. A quadratic boundary with correlated, normally distributed predictor variables
3. A quadratic boundary with independent, non-normally distributed predictor variables.

The complexity of the decision boundaries increases from the first to the third scenario. Because the underlying assumptions for both logistic regression and discriminant analysis are met for the
first two scenarios, these stable methods are expected to have superior predictive accuracy to boosted trees. However, in the third scenario, various assumptions of the stable classification methods will be violated, giving boosted trees the opportunity to show superior performance.

Within scenarios, the influence of several factors will be investigated including (1) the size of the training data set; (2) the complexity of the base classifier used in boosting; and (3) the degree of distributional overlap between cases and controls. By investigating the influence of these parameters, I hope to clarify which factors make the most important contributions to optimal performance for both the boosted trees and the two stable classification methods.

Thus, my research has two main hypotheses.

H1: Boosted classification trees will have prediction error rates (i.e., as measured by the overall misclassification rate for the test set) that are lower than single classification trees for all of the scenarios tested.

H2: Discriminant analysis and logistic regression will have prediction error rates that are lower than that of boosted classification trees when the underlying assumptions for the two stable methods are being met. In contrast, boosted classification trees will have lower prediction error rates than the two stable methods when the underlying assumptions of the stable methods are being violated.

A secondary objective of this research is to determine whether boosting improves the accuracy of predictions made by logistic regression. While Breiman’s research suggests that boosting will be unproductive for logistic regression, this suggestion has not yet been directly demonstrated in the literature. This objective leads to my final hypothesis.

H3: Boosting will not improve the predictive accuracy of logistic regression.
Methods

All simulations and statistical analyses, with the exception of power calculations, were conducted using R 2.4.1 for Windows. The R Project offered program packages for conducting each of the three classification methods used in this research, as well as a program for boosting classification trees. Because the source code was available for these programs, the programs could be adapted to meet the needs of this research.

The methods for this research are described in five sections. The first section describes each of the three classification methods and the key features of the R programs used to implement them. The second section describes the boosting algorithm and the R program used to implement it. The third section describes the three classification problems that were used to compare the predictive accuracy of boosted trees, LR, and discriminant analysis. The fourth section describes how the data simulations were performed. The final section describes the statistical tests used to analyze the data and the power calculations that were used to size the study.

The Three Classification Methods

Classification Trees

All classification trees were constructed using the R package, rpart (i.e., recursive partitioning), which employs the CART (Classification and Regression Tree) algorithm developed by Breiman et al. A classification tree begins with a “root” node that includes the training data for both cases and controls. The CART algorithm then partitions the root node into two “child” nodes by identifying the one predictor variable that best discriminates between cases and controls (Figure 1). The child node that has the highest proportion of controls is designated as belonging to
controls \((Y = 0)\), while the child node that has the highest proportion of cases is designated as belonging to cases \((Y = 1)\). For a one-level tree (also known as a stump) no further splits are attempted and the two child nodes are also terminal nodes (i.e., nodes with no further splits).

For a two-level tree, the first two child nodes are each further partitioned into two child nodes using the predictor variable that best discriminates between cases and controls within each original child node (Figure 1). Thus, a two-level classification tree can have up to four terminal nodes. Because a stump partitions the training data using only a single predictor variable, stumps are capable of modeling only main effects. In contrast, a branch within a two-level tree can use two predictor variables to discriminate between cases and controls. So, two-level trees are capable of modeling two-way interactions between predictor variables. When classification trees were boosted, only stumps and two-level trees were used as base classifiers because the classification problems chosen did not require terms higher than second order.

The \textit{rpart} package allows users to control various aspects of a classification tree’s structure including: (1) the maximum depth of nodes in the final tree (\textit{maxdepth}); (2) the minimum number of observations that must exist in a node before a split is attempted (\textit{minsplit}); and (3) the minimum number of observations that must be present in a terminal node (\textit{minbucket}). Additionally, \textit{rpart} allows control over the complexity parameter (\textit{cp}), a goodness-of-fit parameter that influences the extent to which trees will grow (or, if desired, the extent to which complex trees will be pruned).

When using stumps as base classifiers, the \textit{rpart} parameters were set as follows: \textit{maxdepth} = 1 (i.e., only 1 level beyond the root node was allowed, thus creating a stump),
Figure 1. Examples of one-level (stumps) and two-level classification trees. The training data set had 1,000 cases (Y=1) and 1,000 controls (Y=0) and the set of predictor variables was ten-dimensional with continuous variables (X₁, X₂, …, X₁₀). The overall misclassification rate for the stump was 0.331 (662/2,000) while the misclassification rate for the more complex two-level tree was slightly lower at 0.290 (579/2,000).
\( \text{minsplit} = 1 \) – the total number of observations, and \( \text{minbucket} = 1 \). These same parameters were used whether large (\( n = 2,000 \)) or medium-sized (\( n = 200 \)) training data sets were being fit. When two-level trees were used as base classifiers, the \( \text{rpart} \) parameters were set as follows: \( \text{maxdepth} = 2; \text{minsplit} = 200 \) (large data sets) or 50 (medium data sets); and \( \text{minbucket} = 1 \). For both stumps and two-level trees, the complexity parameter was set equal to zero. Because both stumps and two-level trees have simple structures, there was no need to use a complexity parameter to limit their growth.

Because constructing a single, multi-level tree requires much less computational effort than constructing a boosted tree, single multi-level classification trees were created as benchmarks. The \( \text{rpart} \) program allows the creation of classification trees that have up to 30 levels. Because such trees are unnecessarily complex for most applications, a non-zero value of the complexity parameter (0.005) was used to limit tree growth. This value was one-half of the smallest difference in test error rates between groups that was to be detected as statistically significant. Thus, any binary split that did not decrease the overall lack of fit by 0.005 was not attempted. For both large and medium-sized data sets, the values of \( \text{maxdepth} \) and \( \text{minsplit} \) were set at 30, and 10, respectively, giving \( \text{rpart} \) maximum flexibility to create complex trees within the boundaries set by the complexity parameter.

**Logistic Regression**

The \( \text{glm} \) function (generalized linear models) of the R package, \textit{stats}, was used to perform logistic regression (LR) analyses. When the binomial distribution was specified as the error function and the logit was used as the model link function, \( \text{glm} \) performed an LR analysis on the specified model. The \( \text{glm} \) function also accepted weights for each of the observations, making
the function sufficiently flexible to be used in the boosting algorithm. Depending upon the
classification problem being investigated, one of three types of logistic models was specified:
1) a model that included only main effects; 2) a model that included only quadratic terms (i.e.,
each predictor variable was squared); and 3) a model that included a combination of main effects
and quadratic terms. The latter model type was used in the most complex classification problem
to investigate the influence of model misspecification on the predictive accuracy of LR.

**Discriminant Analysis**

Discriminant analysis fits a boundary between classes by maximizing the ratio of the between-
class variance to the within-class variances. When the populations of cases and controls are
multivariate normal and share the same dispersion matrix, linear discriminant analysis (LDA) is
used to make class assignments. Under LDA, the *ith* observation of predictor variables, *x_i*, is
allocated as a case (*Y = 1*) if the following criterion is met:

\[
\mu_1 - \mu_0 \Sigma^{-1} x_i - 0.5 \ast (\mu_1 - \mu_0) \Sigma^{-1} (\mu_1 + \mu_0) \geq \ln \left( \frac{c(0|1)p_0}{c(1|0)p_1} \right)
\]

Here, \( \mu_1 \) and \( \mu_0 \) represent the vector of means for the population of cases and controls,
respectively, and \( \Sigma^{-1} \) represents the inverse of their common dispersion matrix. The terms, \( c(0|1) \)
and \( c(1|0) \), represent the respective misclassification costs for cases and controls. The terms \( p_1 \)
and \( p_0 \) represent the respective prior probabilities for cases and controls. Notice that the above
equation is linear with respect to \( x_i \), making the boundary between classes a linear one (i.e., a
hyperplane) and amenable to analysis by LDA.

When developing diagnostic tests, the misclassification costs would be determined by
considering the relative costs of false positives and false negative tests for the disease of interest.
Likewise, the uninformative prior probabilities would be fixed by the prevalence of disease in
the population of interest. Because this research was of a theoretical nature, it was appropriate to assume equal misclassification costs for controls and cases and that the proportion of cases and controls were equal. Thus, the equation for allocating observations as cases under LDA becomes:

$$(\mu_1 - \mu_0)' \Sigma^{-1} x_i - 0.5*(\mu_1 - \mu_0)' \Sigma^{-1} (\mu_1 + \mu_0) \geq 0 \quad (2)$$

In situations where the populations of cases and controls do not share the same dispersion matrix (i.e., where $\Sigma_1 \neq \Sigma_0$), quadratic discriminant analysis (QDA) minimizes misclassification errors by assigning the observation, $x_i$, as a case ($Y = 1$) if the following criterion is met:

$$-0.5* x_i'(\Sigma_1^{-1} - \Sigma_0^{-1}) x_i + (\mu_1' \Sigma_1^{-1} - \mu_0' \Sigma_0^{-1}) x_i > 0 \quad (3)$$

Again, equation 3 assumes equal misclassification costs for cases and controls, as well as equal prior probabilities for the two classes. Notice that the allocation rule is a quadratic function of $x_i$, making the boundary between classes a quadratic one, rather than a linear one. In such cases, QDA, rather than LDA, was the appropriate classification method for analysis.

LDA and QDA were performed using the *lda* and *qda* functions of the R package, *MASS*. The prior probabilities of class membership were left unspecified. This choice meant that the *lda* and *qda* programs estimated these probabilities using the class proportions in the training set, which was 0.5 for both cases and controls.

**Boosting Algorithm**

A variety of algorithms are available for boosting, each of which uses its own method for reweighting the training data and for weighting the linear combination of classifiers.\textsuperscript{26} For the purposes of this research, the *Real AdaBoost* algorithm was used. Friedman et al. have shown that, for a variety of data sets, *Real AdaBoost* offers high predictive accuracy and a rapid
convergence to a minimum error rate.\textsuperscript{26} Additionally, \textit{Real AdaBoost} does not involve bagging, which would slow computation if included. Friedman et al. have shown that bagging is not required to achieve increased prediction accuracies from boosting.\textsuperscript{26}

\textit{Real AdaBoost} works as follows. Suppose that the observations in the training data set are \((y_i, x_i)\), where \(x_i\) is the vector of \(p\) predictor variables and \(y_i\) is the observed class label for the \(ith\) observation \((i = 1, 2, \ldots, N)\). In the first step of our boosting process, the training data are fit using the base classifier of choice, giving each observation equal weight, \(w_i = 1/N\). Using this fit, the algorithm calculates the probability that an observation is a case \(\{p_m(x_i) = P(Y = 1 | x_i)\}\).

Here, \(m\) is the index for the current iteration of the boosting process. The classifier, \(f_m(x_i)\), is calculated as

\[
    f_m(x_i) = 0.5 \cdot \log \left\{ \frac{p(x_i)}{(1 - p(x_i))} \right\}, \hspace{1cm} (4)
\]

where \(f_m(x_i)\) can range from \(-\infty\) to \(+\infty\). The sign of \(f_m(x_i)\) indicates the class allocation made by the current classifier, with +1 and –1 indicating assignment to cases and controls, respectively. Thus, observations are allocated as cases if \(p(x_i) > 0.5\) for the current version of the training data set (i.e., the original data set in boosting step 1).

Next, the class probabilities and the classifiers produced are used to reweight the data for the next round of fitting. Weights for each observation are recalculated as

\[
    w_i = w_i^* \cdot \exp[-y_i f_m(x_i)], \hspace{1cm} (5)
\]

and the weights are then renormalized so that their sum equals unity. When the class prediction of the current classifier, \(f_m(x_i)\), matches the observed class for an observation, that observation’s weight is decreased for the next fitting step. The extent to which the observation is down-weighted increases as the probability with which the classifier assigns that observation as being a case increases. When the class prediction of the current classifier disagrees with the observed
class for an observation, that observation’s weight is increased for the next fitting step. The extent to which the observation is up-weighted increases as the probability with which the classifier assigns that observation as being a case increases. The net effect of the reweighting process is to create a modified version of the training set in which observations that are misclassified by the current classifier are emphasized.

This process of refitting the reweighted data, calculating new class probabilities and classifiers, and using the current classifier to reweight the data continues for a total of \( m \) boosting steps. A summary of the Real AdaBoost algorithm, adapted from Friedman et al.\(^{26} \), is shown in Figure 2.

---

**Real AdaBoost**

1. In the first step, use weights \( w_i = 1/N, i = 1, 2, \ldots, N \), where \( N \) is the total number of observations in the training data set.

2. Repeat for \( j = 1, 2, \ldots, m \), where \( m \) is the total number of boosting steps to be performed.
   
   (a) Fit the chosen base classifier (e.g., classification trees) to obtain a class probability estimate, \( p_j(x_i) = P(y_i = +1 \mid x_i) \), using the weights \( w_i \) on the training data.

   (b) Set the classifier for step \( j \), \( f_j(x_i) = 0.5\log[p_j(x_i)/(1 - p_j(x_i))] \)

   (c) Set \( w_i = w_i*\exp[-y_if_j(x_i)], i = 1, 2, \ldots, n \). Then, renormalize the weights such that \( \sum_i w_i = 1 \).

3. Output the classifier sign \{i.e., sign of \( F(x) = \sum_j f_j(x) \)\}. Note that for this algorithm, each of the \( m \) classification rules is given equal weight.

---

Figure 2. Summary of the Real AdaBoost algorithm. Adapted from Friedman et al.\(^{26} \)
For reasons of cost and programming flexibility, the existing R package developed by Marcel Dettling named *boost* was used to implement the boosting performed in this research. While Dettling’s R package did not offer a module to implement the *Real AdaBoost* algorithm directly, his *adaboost* module, which implements the closely related *Discrete AdaBoost* algorithm, was modified to meet the needs of this research.

The key modifications made to Dettling’s program included the following: (1) using the probabilities that observations were cases (i.e., \(P(y_i = 1 \mid x_i)\)) to construct classifiers rather than using the predicted classifications themselves; (2) giving each classifier in the boosting procedure equal weight; (3) allowing classification trees other than stumps to be used in the boosting procedure; and (4) allowing an LR model to be used as a base classifier in the boosting procedure. The R code for this modified program is shown in Appendix 1.

Unlike commercially available boosting programs, Dettling’s program lacked any automatic feature to prevent overfitting the data during the boosting procedure. However, this feature was not needed for this research, which involved simulated data. Large test sets (\(n = 2,000\)) were generated and the minimum value of the misclassification rate for the test set could be easily identified. It was the minimum values of the test error rates that were used to compare the predictive accuracy of the three classification methods.

### The Three Classification Scenarios

Three classification problems or scenarios were selected to compare the predictive accuracy of the classification methods, with each scenario offering a potential performance advantage to one or two of the methods. Each classification problem was binary, and the underlying populations had equal numbers of cases (\(Y=1\)) and controls (\(Y=0\)). Each scenario also used 10 predictor
variables, with each predictor variable being relevant to the prediction of the binary outcome. Because each scenario used a minimum of 10 training observations per predictor variable per class, all comparisons were performed under conditions where both discriminant analysis and LR behaved as stable classification methods. The three scenarios, which will be discussed below, are described by the type of decision boundary involved (linear or quadratic) and by the underlying distributions of the populations of cases and controls (normal or non-normal). The three scenarios included: (1) a linear boundary from normal populations; (2) a quadratic boundary from normal populations; and 3) a quadratic boundary from non-normal populations.

Scenario 1: Linear Boundary, Normally Distributed Populations

In Scenario 1, the population of controls had a multivariate normal distribution, \( Y_0 \sim N_{10} (0, I) \), where \( 0 \) was a vector of 10 identical means equal to zero, and \( I \) was the 10-dimensional identity matrix. The population of cases had a multivariate normal distribution, \( Y_1 \sim N_{10} (\mu_1, I) \), where \( \mu_1 \) was a vector of 10 identical, non-zero means. To vary the degree of overlap between cases and controls and to produce a range of minimum misclassification rates, three values of \( \mu_1 \) (0.25, 0.50, and 0.75) were used. Because both populations were multivariate normal and shared a common dispersion matrix (\( I \)), this scenario involved a decision boundary that was linear with respect to the vector of predictor variables (i.e., a 10-dimensional hyperplane). Here \( \mu_0 \) and \( \mu_1 \) represent the vector of population means for controls and cases, respectively, and \( \Sigma \) represents the populations’ common dispersion matrix, \( I \). The MMRs calculated for the three selected
values of $\mu_1$ were 0.346 ($\mu_1 = 0.25$), 0.215 ($\mu_1 = 0.50$), and 0.118 ($\mu_1 = 0.75$). These equations assume equal misclassification costs and equal uninformative prior probabilities for cases and controls.

Because the assumptions of LDA are fully met in this scenario (i.e., normal distributions and common dispersion matrix), LDA was expected to yield test set misclassification rates that approached Fisher’s MMR. Likewise, because the predictor variables were uncorrelated and multicollinearity was not a concern, a logistic model that used only main effects was expected to provide test set misclassification rates close to Fisher’s MMR. While boosted trees were also expected to perform well (i.e., the Neyman-Pearson lemma suggests that boosted decision trees will provide an optimal combination of predictors), boosted trees were expected to have test error rates that were somewhat higher than those for LR or LDA because boosted trees produce approximations to the decision boundary.

**Scenario 2: Quadratic Boundary, Normally Distributed Populations**

This scenario involved a quadratic decision boundary that was intended to provide quadratic discriminant analysis (QDA) and boosted classification trees with an advantage in predictive accuracy over LR. As in Scenario 1, the control population had a multivariate normal distribution, $Y_0 \sim N_{10} (0, I)$. Cases had a multivariate normal distribution, $Y_1 \sim N_{10} (\mu_1, \Sigma_1)$, where $\mu_1$ was again a vector of 10 identical but non-zero means. Three values of $\mu_1$ (0.1, 0.5, and 0.9) were chosen to provide a range of minimum misclassification rates. In this scenario, the dispersion matrix for cases ($\Sigma_1$) was defined such that predictor variables 1 through 3 had a common covariance equal to 0.3 and predictors 4 through 6 had a common covariance equal to
0.5. Predictors 7 through 10 were retained as independent variables. The upper half of the symmetric matrix $\Sigma_1$ is shown below.

$$
\Sigma_1 = 
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
$$

Because the populations of cases and controls did not share a common dispersion matrix ($\Sigma_1 \neq \Sigma_0$), the decision boundary between classes was quadratic with respect to the vector of predictor variables for this scenario (see equation 3). Because the distributional assumptions for QDA are met under Scenario 2, QDA should produce test error rates that are close to the minimum misclassification rates.

The misclassification rates for boosted trees will depend upon the type of classification tree used as the base classifier. Boosted stumps were expected to yield higher test error rates than boosted two-level trees because stumps can only model main effects. Because boosted classification trees produce an approximation to the decision boundary, the expected ordering of test set misclassification rates for this scenario was QDA < boosted two-level trees < boosted stumps.

Scenario 2 represented a mild challenge for logistic regression. The correlation between several predictor variables reduced the amount of information in the data set (i.e., as compared to when the predictors were independent). Thus, LR was expected to have higher test error rates than QDA. However, it is unclear how LR’s test error rates will compare to boosted decision
trees. The hypothesis for this research was that LR would yield test error rates higher than those of boosted two-level trees but similar to those of boosted stumps.

The Bayes error rate was used to estimate the minimum misclassification risk for this scenario. The Bayes error rate is simply the error rate associated with the Bayes classifier, which assigns observations to classes on the basis of their posterior probabilities. Because the proportion of cases and controls were equal in these simulations, the use of uninformative priors reduced the Bayes classifier to the Bayes factor, BF(Y) = P(X|Y1)/P(X|Y0). Thus, if BF(Y) > 1, the Bayes classifier assigned the observation as a case (Y = 1); otherwise, the observation was classified as a control (Y = 0).

Numerical integration was used to estimate the Bayes error rate. The MASS package of R was used to randomly generate 10,000 multivariate observations from the population of controls (Y0) and 10,000 multivariate observations from the population of cases (Y1). Then, using the R package mvtnorm, the probability density for each multivariate observation was calculated assuming Y0 and Y1, and each observation was classified as being either a case or a control according to the value of the Bayes classifier. The total proportion of misclassified observations provided an estimation of the Bayes (minimum) error rate. This process was repeated 10 times and these values were averaged to obtain a standard deviation for the Bayes error rate. The R program used to calculate the Bayes error rate is shown in Appendix 2. The Bayes error rates for the three values of μi used in Scenario 2 were 0.320 ± 0.0031 (μi = 0.1), 0.198 ± 0.0024 (μi = 0.5), and 0.084 ± 0.0016 (μi = 0.9).
Scenario 3: Quadratic Boundary, Non-Normally Distributed Populations

Scenario 3 used Friedman’s nested spheres decision boundary, a complex decision boundary that uses a quadratic function of the predictor variables to assign observations to classes. The observations for the test and training sets were created as follows. First, a total number of observations, equal to the total number of cases and controls, was generated with each observation having a standard multivariate normal distribution, \( Y \sim N_{10}(0, I) \). Then, the values of the individual predictor variables within an observation were squared and summed (i.e. \( \sum_i x_i^2 \), where \( i = 1, 2, \ldots 10 \)). Observations having values of \( \sum_i x_i^2 > 9.34 \) (i.e., the value of chi-square distribution, \( \chi^2_{0.5} (10) \)) were assigned as cases. Otherwise, observations were assigned as controls. This process created two ten-dimensional spheres that were separated by a threshold value. Because the class assignments were made in a deterministic manner, the minimum (Bayes) error rate for this scenario was equal to zero.

Because class assignments were made using a chi-square distribution, the distributional assumptions for QDA were violated. Thus, QDA was expected to provide test error rates that were higher than the minimum error rates in this scenario. Likewise, the predictive accuracy of LR was expected to depend upon the degree to which the logistic model was properly specified. LR models that used only main effects would produce high test error rates. LR models that were properly specified (i.e., used quadratic functions of each predictor) were expected to produce the lowest test error rates because they used the same deterministic formula that was used to generate the data. LR models with varying degrees of misspecification (from 1 to 10 predictors could be misspecified) would yield intermediate error rates.

Because two-level trees could model quadratic terms, it was expected that boosted two-level trees would produce low test error rates, probably similar to those produced by QDA. It
was hypothesized that boosted stumps would provide test error rates intermediate to those of QDA and improperly specified logistic regression models.

The hypothesized orderings of the test error rates for each classification method are summarized in Figure 3 for each scenario.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Decision Boundary/Populations</th>
<th>Ordering by Hypothesized Test Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Linear/Normal</td>
<td>LDA ≈ LR &lt; boosted stumps ≈ boosted two-level trees &lt; single complex tree</td>
</tr>
<tr>
<td>2</td>
<td>Quadratic/Normal</td>
<td>QDA &lt; boosted two-level trees ≈ LR &lt; boosted stumps &lt; single complex tree</td>
</tr>
<tr>
<td>3</td>
<td>Quadratic/Non-Normal</td>
<td>LR(quadratic) &lt; QDA ≈ boosted stumps &lt; boosted 2-level trees &lt; single complex tree &lt; LR(linear)</td>
</tr>
</tbody>
</table>

Figure 3. Hypothesized Ordering of Test Error Rates by Classification Method

**Conducting the Simulation Studies**

The simulation studies involved applying variants of the three classification methods to a series of independently generated training and test data sets. These variants included:

1. Boosted stumps
2. Boosted two-level trees
3. A single, complex classification tree
4. Boosted LR (using linear, quadratic, or combination models)
5. LDA or QDA, as appropriate

An overview of the simulation process is shown in Figure 4. The first step in a simulation was to select the particular scenario and condition to be investigated. There were three conditions...
each for Scenario 1 and Scenario 2, corresponding to the three different choices for \( \mu_1 \) for those scenarios. Scenario 3 involved only a single condition. In addition, each scenario/condition combination was tested using both large \((n = 2,000)\) and medium-sized \((n = 200)\) training sets.

Next, independent training and test data sets were generated from the populations specified in the selected scenario/condition. Training sets were either large \((n = 2,000)\) or of medium size \((n = 200)\) and had observations that were evenly divided between cases and controls. Test data sets always contained a total of 2,000 observations with equal numbers of cases and controls. R program modules were written to generate the training and test data for each scenario. A copy of these data generation modules can be found in Appendix 3.

The next step in the simulation was to determine the misclassification rate for the test data set using the first classification method, boosted stumps. Because sensitivity and specificity were regarded as equally important for this research, the test error rate was simply calculated as the total proportion of test data that were misclassified. The test error rate was calculated at every boosting iteration and saved to a file for later use. A total of 200 boosting steps was performed for all boosted methods because a series of preliminary runs had shown that all the classification methods reached a minimum test error rate within 200 boosting steps.

Once the test error rates had been calculated for boosted trees, new training and test sets were randomly generated. Then, the next of the five classification methods, boosting with two-level trees, was applied. Again, the test error rate was calculated at each boosting iteration and the data were saved for later use. This process was repeated until all five variants of the classification methods had been applied, each to its own independent training and test data sets.
1. Select scenario/condition (e.g., Scenario #2, \( \mu_1 = [0.1, 0.1, \ldots, 0.1] \) )

2. Set the number of boosting steps to be performed at \( m = 200 \)

BEGIN RUN #1

3. Generate independent training data (\( n = 2,000 \) or 200) and test data (\( n = 2,000 \)) sets

4. Perform **boosting with stumps**, calculating/saving the misclassification rate for the test set at each boosting step (for 200 boosting steps)

5. Generate new, independent training and test sets

6. Perform **boosting with two-level trees**, calculating/saving the misclassification rate for the test set at each boosting step (for 200 boosting steps)

7. Generate new, independent training and test sets

8. Fit a **single, complex classification tree** to the training data. Calculate the misclassification rate for the test set

9. Generate new, independent training and test sets

10. Perform **boosting with logistic regression**, calculating/saving the misclassification rate for the test set at each boosting step (for 200 boosting steps)

11. Generate new, independent training and test data sets

12. Fit the training data using **LDA (or QDA)**. Calculate the misclassification rate for the test set.

REPEAT STEPS 3 – 11 for a TOTAL OF 90 RUNS

13. Average and plot the data (i.e., as a function of the number of boosting steps).

14. Create permanent data files saving the misclassification rates for each method.

Figure 4. Summary of the steps performed in the data simulations.

This entire process was repeated 90 times. Thus, a total of 90 training and 90 test sets were fit by each of the five classification methods. For large training sets, this meant that a total of 180,000 training observations and 180,000 test observations were examined by each
classification method. For medium-sized training sets, a total 18,000 training observations and 180,000 test observations were examined by each method (i.e., test set sizes were unchanged). The misclassification rates for the test set data generated by each method were saved at each step of the boosting process and for all 90 test methods examined. These data were used for subsequent statistical comparisons of the accuracy of the five methods.

**Statistical Analyses and Power Calculations**

A series of $z$-tests were conducted to compare the test set misclassification rates between the classification methods. Within a particular scenario/condition, a total of 13 inferential tests were performed. Three of these tests assessed whether boosting significantly reduced the misclassification rates when using various base classifiers (Hypotheses 1 and 3). Ten of these tests involved two-way comparisons of the minimum misclassification rates achieved by each of the five classification methods (Hypothesis 2). While some of these hypotheses were one-sided, all $z$-tests were performed as two-sided tests and used a Bonferroni adjustment for the multiple comparisons. This approach was conservative, increasing the likelihood that very small differences in the test error rates between methods would not be detected.

Power calculations were performed using the two-sample frequency function of PROC POWER (SAS 9.1). The calculation showed that a total of 174,040 observations (equally divided between cases and controls) were needed to have a 90% power of detecting a difference of 0.01 between two misclassification rates (where the reference proportion equaled 0.5) using a two-sided $\alpha = 0.003846$ (i.e., 0.05/13). Thus a total of 180,000 test set observations were used, equally divided between cases and controls.
Results

This research investigated three simulation scenarios, each of which was intended to emphasize the advantages of one or more of the selected classification methods when different types of decision boundaries were encountered. To facilitate the presentation of results, this section is divided into three main parts, each one corresponding to the three types of decision boundaries examined. Because sensitivity and specificity were considered of equal importance for this work, the overall proportion of misclassification in the test set was used as the measure for all method comparisons and, for convenience, is referred to as the test error rate.

Scenario 1: Linear Boundary, Normal Populations

In Scenario 1, the population of both cases and controls were normally distributed and shared a common dispersion matrix, making the decision boundary between the two classes a linear one (i.e., a 10-dimensional hyperplane). Because the underlying assumptions of LDA were fully met here, the application of LDA to the training data were expected to yield test error rates that were close to Fisher’s minimum misclassification rate (MMR). The value of Fisher’s MMR depended upon the separation between classes and, for the three instances examined here, the MMR values were 0.346 ($\mu_1 = 0.25$), 0.215 ($\mu_1 = 0.50$), and 0.118 ($\mu_1 = 0.75$). Because there were a minimum of 20 observations per predictor variable for each class, LR was expected to provide efficient estimates of the model parameters and yield test error rates that were close to those of LDA.48 Boosted classification methods using the unstable classifiers produce only approximations to the decision boundary and were expected to yield test error rates higher than either LDA or LR.
Large Training Sets (n = 2,000)

The results of applying the various classification methods to the large training sets when $\mu_1 = 0.25$ are given in Figure 5. The figure shows the test error rate at each step of the boosting process when stumps, two-level trees, and LR were used as the base classifiers. Figure 5 also shows, as reference lines, the test error rates produced by LDA and by the use of a single, complex classification tree. Each point on the plot is the mean of error rates from 90 independent test sets, each having 2,000 test observations. The mean standard errors for these data had approximately the same thickness of the lines in the plots (i.e., $\approx 0.00002$), and as a consequence, this and similar figures do not contain error bars.

As Figure 5 illustrates, boosting improved the predictive accuracy of the two unstable classifiers—stumps and two-level trees. When stumps were used as the base classifiers, the test error rate fell from 0.452 to 0.375 after 43 boosting steps ($P < 0.0001$). After reaching this minimum value, the test error rate continued to increase gently, suggesting a slight amount of overfitting when more classifiers were added to the ensemble. When two-level trees were used as the base classifiers, the test set error rate dropped from 0.443 to 0.380 after 11 boosting steps ($P < 0.0001$). After reaching this minimum value, the test error rate increased as boosting proceeded, suggesting that overfitting occurred once 11 two-level trees had been added to the ensemble (Figure 5). While the minimum test error rates for boosted stumps and boosted trees were similar (i.e., 0.375 vs. 0.380), this small difference was statistically significant ($P < 0.0001$) due to the large number of test observations used in the analysis.

When LR was used as the base classifier, boosting had no effect on the minimum test error rate (Figure 5). The test error rate was nearly constant at 0.347 ($\pm 0.0003$) over all 200 boosting steps. The fact that boosting was unproductive with this stable base classifier was
Figure 5. Scenario 1: $\mathbf{\mu}_1 = (0.25, 0.25, \ldots, 0.25)$ for large training sets ($n = 2,000$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for classification by stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for LDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of Fisher’s minimum misclassification rate for this condition equals 0.346.
consistent with previous research, which showed that boosting was also unproductive with other stable classifiers, such as LDA. As expected, the minimum test error rate for LR was quite similar to Fisher’s MMR (0.346). The mean minimum test error rate of LDA (0.347) was not significantly different from that of LR \( (P = 1) \) and was, again, very close to Fisher’s MMR.

The mean test error rate for a single, complex tree (0.414) was significantly lower than for either a single stump (0.452) or a single two-level tree (0.453) \( (P < 0.0001 \) for both comparisons). This result suggested that having a tree with increased complexity might be useful when only a single classifier, rather than an ensemble of classifiers, was used to make predictions. However, the single complex tree had a test error rate (0.414) that was significantly higher than the minimum test error rate for either boosted stumps (0.375) or boosted two-level trees (0.380). Thus, the ordering of the minimum test error rates for the five classification methods increased as follows (Table 1):

\[
\text{LDA} = \text{LR} < \text{boosted stumps} \approx \text{boosted trees} < \text{single complex tree}.
\]

The same patterns were observed when the classification methods were applied to large training sets where the population of cases had less distributional overlap with the controls. The results for applying the various classification methods to large training sets for larger values of \( u_1 \) are shown in Figure 6 \( (u_1 = 0.50) \) and Figure 7 \( (u_1 = 0.75) \).

Boosting again improved the test error rates for both of the unstable classifiers—stumps and two-level trees. When boosting stumps, the test error rate decreased from 0.403 to 0.248 at \( u_1 = 0.50 \) \( (P < 0.0001) \) and from 0.358 to 0.152 at \( u_1 = 0.75 \) \( (P < 0.0001) \). When boosting two-
Table 1. Comparison of Minimum Test Error Rates by Classification Method: Scenario 1 with Large Training Sets

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>Value of $\mu_1$</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.25</td>
<td>0.50</td>
<td>0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Error Rate</strong></td>
<td><em><em>Stat.</em> Group</em>*</td>
<td><strong>Error Rate</strong></td>
<td><em><em>Stat.</em> Group</em>*</td>
<td><strong>Error Rate</strong></td>
</tr>
<tr>
<td>Single, Complex Tree</td>
<td>0.414</td>
<td>A</td>
<td>0.325</td>
<td>A</td>
<td>0.246</td>
</tr>
<tr>
<td>Boosted Two-Level Trees</td>
<td>0.380</td>
<td>B</td>
<td>0.251</td>
<td>B</td>
<td>0.151</td>
</tr>
<tr>
<td>Boosted Stumps</td>
<td>0.375</td>
<td>C</td>
<td>0.248</td>
<td>B</td>
<td>0.152</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.347</td>
<td>D</td>
<td>0.217</td>
<td>C</td>
<td>0.119</td>
</tr>
<tr>
<td>LDA</td>
<td>0.347</td>
<td>D</td>
<td>0.217</td>
<td>C</td>
<td>0.119</td>
</tr>
<tr>
<td>Fisher’s MMR§</td>
<td>0.347</td>
<td>---</td>
<td>0.215</td>
<td>---</td>
<td>0.118</td>
</tr>
</tbody>
</table>

*Within a column, methods in different statistical groups had misclassification rates that were significantly different at $\alpha = 0.05$. (No between-column comparisons were tested for significance)

§ Fisher’s minimum misclassification rate (see Methods section).

As the boundary between cases and controls became more distinct (i.e., $\mu_1$ increased), the test error rate decreased from 0.386 to 0.251 at $\mu_1 = 0.50$ ($P < 0.0001$, Figure 6) and from 0.331 to 0.151 at $\mu_1 = 0.75$ ($P < 0.0001$, Figure 7). However, boosting was again unproductive when applied to LR, with the test error rate for LR remaining nearly constant at 0.217 for $\mu_1 = 0.50$ and at 0.119 for $\mu_1 = 0.75$ for all boosting steps (see Figures 6 and 7).

As the boundary between cases and controls became more distinct (i.e., $\mu_1$ increased), the boosting profiles for the stumps and two-level trees became more similar (e.g., compare Figures 5 and 7). The small difference in minimum test error rates between boosted stumps and
Figure 6. Scenario 1: $\mu_1 = (0.50, 0.50, \ldots, 0.50)$ for large training sets ($n = 2,000$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for classification by stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for LDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of Fisher’s minimum misclassification rate for this condition equals 0.215.
Figure 7. Scenario 1: $\mu_1 = (0.75, 0.75, \ldots, 0.75)$ for large training sets ($n = 2,000$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for classification by stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for LDA and for a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of Fisher’s minimum misclassification rate for this condition equals 0.118.
boosted two-level trees was no longer statistically significant at either $u_1 = 0.50$ or $0.75$ (Table 1). Likewise, the degree of overfitting for two-level trees also decreased as $u_1$ increased, with the boosting profiles of stumps and two-level trees becoming virtually identical at $u_1 = 0.75$ (Figure 7). This pattern suggested that the boosting procedure’s tendency to overfit the data may be reduced when the boundary between cases and controls becomes clearer.

The minimum test error rates for all five methods decreased as $u_1$ increased from 0.25 to 0.75 (compare Figures 5 and 7). However, the relative orderings of the minimum test error rates by classification method were unchanged (Table 1). LDA and LR had the lowest test error rates of all the classification methods for all values of $u_1$. The test error rates of LDA and LR were not significantly different at $\alpha = 0.05$ for any value of $u_1$ and these rates approached Fisher’s MMR for all values of $\mu_1$ (Table 1). Boosted stumps and boosted two-level trees had test error rates intermediate to LDA/LR and a single, complex classification tree. Thus, the two stable classification methods produced the lowest test error rates for the linear boundary of this scenario under all three values of $\mu_1$. The boosted classifiers provided error rates that were somewhat higher than the stable classification methods, while the single complex tree had the highest error rate. This single, complex tree performed poorly relative to the boosted classifiers and to the stable classifiers, even though a complex tree contained from 12 to 20 terminal nodes.

**Medium-Sized Training Sets ($n = 200$)**

The boosting profiles for the five classification methods trained on medium-sized training sets are shown in Figure 8 ($u_1 = 0.25$), Figure 9 ($u_1 = 0.50$), and Figure 10 ($u_1 = 0.75$). The most salient difference between using large and medium-sized training sets was that the
Figure 8. Scenario 1: $\mathbf{\mu}_1 = (0.25, 0.25, \ldots, 0.25)$ for medium-sized training sets ($n = 200$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for classification by stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for LDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of Fisher’s minimum misclassification rate equals 0.346.
Figure 9. Scenario 1: $\mathbf{\mu}_1 = (0.50, 0.50, \ldots, 0.50)$ for medium-sized training sets ($n = 200$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for classification by stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for LDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of Fisher’s minimum misclassification rate equals 0.215.
Figure 10. Scenario 1: $\mu_1 = (0.75, 0.75, \ldots, 0.75)$ for medium-sized training sets ($n = 200$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for LDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of Fisher’s minimum misclassification rate equals 0.118.
minimum test error rates for all the classification methods were higher for the medium-sized training sets (e.g., compare Figures 5 and 8). This finding suggested that all the classification methods produced less precise estimates of the decision boundary when they had fewer observations of each class on which to train.

The other notable difference between using the large and the medium-sized training sets was that the gap in the minimum error rates between boosted stumps and boosted trees became more pronounced, especially at the lowest value of $u_1$ (e.g., compare Figures 5 and 8). However, this difference again disappeared at the highest value of $u_1$ (Figure 10).

The same ordering of minimum test error rates by classification method was seen for both large and medium-sized training sets (Table 2). Boosting improved the predictive accuracy of weak classifiers but not of the stable classifier, LR. The minimum error rates for LR and LDA were not significantly different, although these values were now further from Fisher’s MMR than with the large data sets (i.e., 0.015 v. 0.001 unit difference). Boosted stumps and trees had error rates that were intermediate to those of LDA/LR and to that of a single, complex tree. Thus, even though the medium-sized training sets contained less information about this linear decision boundary than the large training sets, this reduction in information did not alter the relative predictive accuracy of the classification methods investigated.
Table 2. Comparison of Minimum Test Error Rates by Classification Method: Scenario 1 with Medium-Sized Training Sets

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>Value of $\mu_1$</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.25</td>
<td>0.50</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>Error Rate</td>
<td>Stat.*</td>
<td>Group</td>
</tr>
<tr>
<td>Single, Complex Tree</td>
<td>0.445 A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boosted Two-Level Trees</td>
<td>0.419 B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boosted Stumps</td>
<td>0.402 C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.370 D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>0.366 D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fisher’s MMR§</td>
<td>0.347 ---</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Within a column, methods in different statistical groups had misclassification rates that were significantly different at $\alpha = 0.05$. (No between-column comparisons were tested for significance).

§ Fisher’s minimum misclassification rate (see Methods section).

**Scenario 2: Quadratic Boundary, Normal Distributions**

In Scenario 2, the populations of both cases and controls had multivariate normal distributions. However, the two populations had different dispersion matrices, making the decision boundary between classes a quadratic function of the vector of predictor variables. Because the underlying assumptions of quadratic discriminant analysis (QDA) were met for this scenario, applying QDA to the training data was expected to yield test error rates that were close to the Bayes (minimum) error rate.47

Correlations between predictor variables reduce the amount of information about a decision boundary contained within a set of observations. It should be somewhat more difficult
for the decision boundary to be described precisely by a logistic regression (LR) model. It was expected that the test error rates for LR would be somewhat higher than those observed for QDA in this scenario. Again, because boosted stumps and trees produce approximations to a decision boundary, they were expected to have test error rates that were higher than QDA.

As in the previous scenario, this one investigated the predictive accuracy of the various classification methods at several degrees of distributional overlap between cases and controls. Predictive accuracy was tested at three different values of $u_1$ to produce a range of minimum error rates similar to those of Scenario 1. For the three conditions examined here, the Bayes error rates were 0.320 ($u_1 = 0.1$), 0.198 ($u_1 = 0.5$), and 0.084 ($u_1 = 0.9$).

**Large Training Sets (n = 2,000)**

The results of applying the five classification methods to the large training sets are shown in Figure 11 for $\mu_1 = 0.1$, where there was the greatest overlap between cases and controls. Of the tested classification methods, QDA yielded the lowest test error rate (0.331), a value that was only slightly higher than the Bayes error rate (0.320). This finding is in line with the expectation that QDA would provide a good estimation of this decision boundary, given that the assumptions for applying QDA were met for this scenario.

Of the classification methods that used unstable classifiers, boosted two-level trees had the highest predictive accuracy. While the test error rate for a single two-level tree was initially only slightly better than random guessing (0.464), a total of 60 boosting steps significantly reduced this error rate to a minimum value of 0.374, a decrease of 0.09 units ($P < 0.0001$). In contrast, boosting produced only a small improvement in the predictive accuracy of stumps. A single stump had a mean test error rate of 0.482. After 40 boosting steps, the test error rate for
Figure 11. Scenario 2: $\mu_t = (0.1, 0.1, \ldots, 0.1)$ for large training sets ($n = 2,000$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for QDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of the Bayes error rate equals 0.320.
stump classifiers fell to 0.461 (Figure 11), a decrease of only 0.021 units ($P < 0.0001$). This decrease was about 5 times smaller than that seen for the boosted two-level trees. In fact, of the five classification methods applied, boosted stumps produced the highest test error rate (Table 3). For this more complex decision boundary, it appeared that using a more complex classifier (i.e., two-level trees v. stumps) improved the ability of the ensemble of classifiers to model the boundary. The single, complex tree had a test error rate (0.426) that was intermediate to that of boosted two-level trees and boosted stumps (Table 4).

Table 3.  Comparison of Minimum Test Error Rates by Classification Method:  Scenario 2 with Large Training Sets

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>Value of $\mu_1$</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>0.5</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error Rate</td>
<td>Stat.* Group</td>
<td>Error Rate</td>
<td>Stat.* Group</td>
<td>Error Rate</td>
</tr>
<tr>
<td>Single, Complex Tree</td>
<td>0.426</td>
<td>A</td>
<td>0.305</td>
<td>A</td>
<td>0.197</td>
</tr>
<tr>
<td>Boosted Stumps</td>
<td>0.461</td>
<td>B</td>
<td>0.265</td>
<td>B</td>
<td>0.128</td>
</tr>
<tr>
<td>Boosted Two-Level Trees</td>
<td>0.374</td>
<td>C</td>
<td>0.242</td>
<td>C</td>
<td>0.113</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.447</td>
<td>D</td>
<td>0.236</td>
<td>D</td>
<td>0.098</td>
</tr>
<tr>
<td>QDA</td>
<td>0.331</td>
<td>E</td>
<td>0.205</td>
<td>E</td>
<td>0.085</td>
</tr>
<tr>
<td>Bayes Error Rate</td>
<td>0.320</td>
<td>---</td>
<td>0.198</td>
<td>---</td>
<td>0.084</td>
</tr>
</tbody>
</table>

*Within a column, methods in different statistical groups had misclassification rates that were significantly different at $\alpha = 0.05$. (No between-column tests of significance were performed)
Logistic regression had low predictive accuracy under this set of conditions. Of the five classification methods used, it was superior only to boosted stumps (Table 4). Again, boosting of LR was found to be unproductive; the initial test error rate (0.447) was not decreased by boosting. While the inability of boosting to improve the predictive accuracy of LR was understandable in Scenario 1, it seems surprising here. In Scenario 1, LR produced the optimal combination of predictor variables to model the decision boundary in the very first boosting step. Only observations that could not be properly classified by even the best set of classifiers were misclassified by LR in Scenario 1, leaving no room for improvement by boosting. In Scenario 2, however, LR appeared to be acting as a weak classifier. In the first boosting step, LR had test error rates that ranged from 0.408 to 0.477. So, LR was initially more accurate than random guessing but there were many more observations that could have been classified properly (i.e., the Bayes error rate was 0.320). So why didn’t boosting help improve LR’s accuracy when LR seemed to be acting as a weak classifier?

A closer look at the individual boosting steps provided an explanation for this behavior. When stumps or two-level trees were used as classifiers, only one or two predictor variables were used to classify the data in the first boosting step. At the second boosting step, a new set of one or two predictors were chosen to classify the reweighted data. As boosting proceeded with trees, this process of selecting new variables by which to classify the data continued until all of the predictor variables were represented in the ensemble.

LR behaved in a somewhat different manner. In this particular scenario/condition, LR likewise selected only one or two variables by which to classify the data at the first boosting step (i.e., only one or two variables attained statistical significance in the LR model and these variables had the largest influence on how observations were classified). However, at the second
boosting step, LR did not select a new set of classifiers on which to classify the reweighted data. The values of the coefficients for all 10 predictor variables were reduced approximately ten-fold, suggesting there was no significant association between the reweighted data and any of the predictor variables. So, while LR acted as a weak classifier here (i.e., it had only modest predictive power), it still behaved as a stable classifier. LR apparently extracts all the information that it can extract from the data in the first boosting step and adds no new classifiers as boosting proceeds, making boosting unproductive.

For this scenario/condition, the ordering of test error rate by classification method increased as follows:

QDA < boosted trees < single, complex tree < LR < boosted stumps.

This was quite a different ordering than seen in Scenario 1. Discriminant analysis continued to offer the highest predictive accuracy followed by the classifiers that had a more complex tree structure (i.e., two-level trees and a single, complex tree). LR and boosted stumps had the lowest predictive accuracies for this scenario.

These rankings changed dramatically as the degree of separation between cases and controls increased in Scenario 2. The result of applying the classification methods to the data at values of \( u_1 = 0.5 \) and \( u_1 = 0.9 \) are shown in Figures 12 and 13, respectively. Again, boosting is seen to significantly improve the predictive accuracy of both stumps and two-level trees (\( P < 0.0001 \)). The improvement for stumps was quite pronounced as \( u_1 \) increased above 0.1. The test error rate for stumps was reduced by 0.139 units at \( u_1 = 0.5 \) and by 0.200 at \( u_1 = 0.9 \) (\( P < 0.0001 \) for both differences), compared to the 0.021 unit decrease observed at \( u_1 = 0.1 \). For two-level trees, boosting reduced the test error rate by 0.132 units at \( u_1 = 0.5 \) and 0.170 units \( u_1 = 0.9 \) (\( P < 0.0001 \) for both differences). In contrast, boosting was unproductive when used with LR.
Figure 12. Scenario 2: $\mu_i = (0.5, 0.5, \ldots, 0.5)$ for large training sets ($n = 2,000$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for QDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of the Bayes error rate equals 0.198.
Figure 13. Scenario 2: $\mu_1 = (0.9, 0.9, \ldots, 0.9)$ for large training sets. The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for QDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of the Bayes error rate equals 0.08.
The ranking of the classification methods by test error rate now more closely resembled the rankings seen in Scenario 1 (Table 4). Again, QDA had the lowest test error rate at both $u_1 = 0.5$ (0.205) and $u_1 = 0.9$ (0.085). These error rates approached the corresponding Bayes error rates (0.198 and 0.084) more closely than when $u_1 = 0.1$. However, it was LR, rather than boosted two-level trees, that now followed QDA as the most accurate classifier. The increased separation between cases and controls resulted in notably improved performance for LR. In Scenario 2, the boosted two-level trees always outperformed boosted stumps, suggesting that having a more complex classifier was helpful when trying to model this more complex (i.e., quadratic) decision boundary. Both of these boosted classifiers outperformed the single, complex tree as the extent of separation between the two classes increased.

Medium Training Sets ($n = 200$)

For Scenario 2, the patterns that were seen with the large data sets were repeated in the medium-sized training sets (see Figures 14, 15, and 16). The key difference between the two sets of results was that the test error rates for all five methods increased upon moving from large to medium-sized training sets for all three conditions (e.g., compare Tables 3 and 4). Because identical patterns were seen in the medium-sized and large data sets, no further discussion of these patterns is presented here.
Table 4. Comparison of Minimum Test Error Rates by Classification Method: Scenario 2 with Medium Training Sets

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>Value of $\mu_1$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
<td>0.5</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Error Rate</strong></td>
<td><em><em>Stat.</em> Group</em>*</td>
<td><strong>Error Rate</strong></td>
<td><em><em>Stat.</em> Group</em>*</td>
<td><strong>Error Rate</strong></td>
<td><em><em>Stat.</em> Group</em>*</td>
</tr>
<tr>
<td>Single, Complex Tree</td>
<td>0.465 A</td>
<td>0.347 A</td>
<td>0.236 A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boosted Stumps</td>
<td>0.479 B</td>
<td>0.295 B</td>
<td>0.154 B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boosted Two-Level Trees</td>
<td>0.442 C</td>
<td>0.293 B</td>
<td>0.144 C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.472 D</td>
<td>0.249 C</td>
<td>0.108 D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QDA</td>
<td>0.385 E</td>
<td>0.248 C</td>
<td>0.108 D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bayes Error Rate</td>
<td>0.320 ---</td>
<td>0.198 ---</td>
<td>0.084 ---</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Within a column, methods in different statistical groups had misclassification rates that were significantly different at $\alpha = 0.05$. (No between-column tests of significance were performed)
Figure 14. Scenario 2: $\mu_1 = (0.1, 0.1, \ldots, 0.1)$ for medium-sized training sets ($n = 200$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for QDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of the Bayes error rate equals 0.320.
Figure 15. Scenario 2: $\mu_1 = (0.5, 0.5, \ldots, 0.5)$ for medium-sized training sets ($n = 200$). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for QDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of the Bayes error rate equals 0.198.
Figure 16. Scenario 2: \( \mathbf{\mu}_1 = (0.9, 0.9, \ldots, 0.9) \) for medium-sized training sets (\( n = 200 \)). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression. Mean misclassification rates are also shown for QDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of the Bayes error rate equals 0.084
Scenario 3: Quadratic Boundary, Non-Normal Populations

Scenario 3 provided the most complex decision boundary of the three scenarios examined. The boundary involved two ten-dimensional hyperspheres that were separated by a threshold value which was a quadratic function of the predictor variables. Because the spheres were constructed using the multivariate chi-square distribution rather than the multivariate normal distribution, the assumptions for QDA were not strictly met in this scenario. This violation of the underlying assumptions for QDA was expected to give boosted classification trees a slight advantage in predictive accuracy. Additionally, the model offered an opportunity to see how well LR performed when the logistic model was misspecified. The predictive accuracies of the classification methods were studied for both large and medium-sized data sets. Because the class assignments for cases and controls were made in a deterministic manner, the Bayes error rate for this scenario equaled zero.

Large Training Sets (n = 2,000)

The results of applying the five classification methods to the large training sets are shown in Figure 17. As seen in the two previous scenarios, boosting improved the predictive accuracy of both unstable classifiers. The test error rate for stumps was reduced from 0.472 to 0.347 ($P < 0.0001$) after 53 boosting steps. Boosting produced a much larger improvement in the predictive accuracy for two-level trees, with the test error rate decreasing from 0.429 to 0.098 ($P < 0.0001$) after 182 boosting steps. As seen in Scenario 2, boosting with the more complex unstable classifier (two-level trees) yielded increased predictive accuracy when modeling the quadratic decision boundary. A single complex tree had a test error rate (0.249) that was intermediate to boosted stumps and boosted trees (Table 5).
Figure 17. Scenario 3 for large training sets (n = 2,000). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression (both linear and quadratic models)\(s\). Mean misclassification rates are also shown for QDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of the Bayes error rate equals zero.
Table 5. Comparison of Minimum Test Error Rates by Classification Method: Scenario 3

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>Large Training Set</th>
<th>Medium Training Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error Rate</td>
<td>Stat.* Group</td>
</tr>
<tr>
<td>Log. Reg. (Linear Model)</td>
<td>0.502</td>
<td>A</td>
</tr>
<tr>
<td>Boosted Stumps</td>
<td>0.347</td>
<td>B</td>
</tr>
<tr>
<td>Single, Complex Tree</td>
<td>0.247</td>
<td>C</td>
</tr>
<tr>
<td>Boosted Two-Level Trees</td>
<td>0.098</td>
<td>D</td>
</tr>
<tr>
<td>QDA</td>
<td>0.068</td>
<td>E</td>
</tr>
<tr>
<td>Log. Reg. (Quadratic Model)</td>
<td>0.002</td>
<td>F</td>
</tr>
<tr>
<td>Bayes Error Rate</td>
<td>0.000</td>
<td>---</td>
</tr>
</tbody>
</table>

*Within a column, methods in different statistical groups had misclassification rates that were significantly different at $\alpha = 0.05$. (No between-column tests of significance were performed)

Boosting did not improve the predictive accuracy for LR regardless of the model used. When the model included only main effects, LR yielded a test error rate (0.502) that was no better than random guessing. Consequently, one would not expect that LR would be a usable classifier for boosting; in this case, it provided no discriminatory ability. In contrast, when the logistic model was properly specified by squaring each of the 10 predictor variables, the test error rate (0.002) was very close to the Bayes error rate of zero. Again, boosting would not be expected to provide an improvement in predictive accuracy here. There was virtually no room for improvement after the first boosting step. When the logistic model used all quadratic terms, LR provided the lowest test error rate of the classification methods tested (Table 5).
It was possible to vary the test error rate for a single LR model between zero and 0.5 by varying the number of model terms that were properly specified. Figure 18 illustrates how the test error rate depended upon the number of properly specified terms in the LR model (no boosting). To determine whether boosting would be helpful when an LR model was acting as a weak classifier (i.e., 0 < test error rate < 0.5), boosting was also performed when 2, 4, 6, or 8 model terms were properly specified. In each of these cases, boosting was found to be unproductive (data not shown). Thus, even when LR acted as a weak classifier, boosting was unproductive because the LR classifier was stable, reaching a minimum test error rate in a single boosting step.

It is interesting to compare the performance of the boosted two-level trees with that of LR. Matching the predictive accuracy of boosted two-level trees required properly specifying 9 of the 10 terms in the LR model (see reference line in Figure 18). Thus, in situations where the decision boundary is complex and the investigator has little information about the nature of the proper analytic model to use, boosting with an unstable classifier may provide a high level of predictive accuracy.

Finally, QDA yielded a test error rate (0.068) that was intermediate to LR (quadratic model) and boosted two-level trees. This was the only scenario in which discriminant analysis performed more poorly than LR (P < 0.0001). This likely occurred because this was the only scenario for which the underlying assumption of normality of the populations was not met. Yet, even though this assumption violated, QDA provided predictive accuracy that was superior to boosted classifiers. Thus, in Scenario 3, the test error rates increased in the following order:

LR (quadratic) < QDA < boosted two-level trees < single complex tree < boosted stumps
Figure 18. Dependence of the mean test error rate upon the number of properly specified terms in the LR model for Scenario 3 with large training sets (n = 2,000). The standard deviation of the error rates are approximately the size of the filled-in circles. The test error rate for boosted two-level trees is also shown as a dashed horizontal line.
The test error rate for LR could be made to approach that of any of the other classification methods by altering the number of terms that were properly specified in the logistic model.

**Medium-Sized Training Sets (n = 200)**

The results of applying the five classification methods to training sets of medium size are shown in Figure 19. The ranking of test error rates by method was identical to that seen with the large training sets.

As seen with the two previous scenarios, the key difference between using the medium-sized and the large training sets was that the test error rates for all classification methods were higher with the medium-sized training sets (Table 5, p.54). Another difference was that the gap between LR (quadratic model) and the other four classification methods appeared to increase when moving to the medium-sized training sets. For example, in order to match the predictive accuracy of boosted two-level trees, only 5 or 6 model terms had to be properly specified in the logistic model when using medium-sized training sets (Figure 19). This finding suggested that, under these conditions, LR was the most robust classification method.
Figure 19. Scenario 3 for medium-sized training sets (n = 200). The mean misclassification rate for the test data is shown as a function of the number of boosting steps for stumps, two-level trees, and logistic regression (both linear and quadratic models). Mean misclassification rates are also shown for QDA and a single, complex classification tree (no boosting). Each profile is an average of 90 independent training sets. The value of the Bayes error rate equals zero.
Figure 20. Dependence of the mean test error rate upon the number of properly specified terms in the LR model for Scenario 3 with medium-sized training sets ($n = 200$). The standard deviation of the error rates are approximately the size of the filled-in circles. The test error rate for boosted two-level trees is also shown as a dashed horizontal line.
Discussion

This discussion will be presented in four parts: (1) choosing the best base classifiers; (2) comparing the predictive accuracy of boosted methods with that of stable classification methods; (3) identifying the study’s limitations; and (4) identifying directions for new research.

Choice of Base Classifiers

Before comparing the accuracy of the tested classification methods, it is important to discuss the two features that were characteristic of the base classifiers that performed best upon boosting. First, the classifiers that most benefited from boosting were unstable. That is, these classifiers produced new combinations of predictor variables after the training data were reweighted. Second, the best base classifiers had a level of complexity that matched the complexity of the decision boundary being modeled.

Boosting was productive in this research only when the base classifier was unstable. Boosting improved the predictive accuracy of both stumps and two-level trees under all the scenario/conditions tested. By their simple nature, these classifiers are capable of selecting only a subset of predictors in a single round of fitting. For example, a stump can use only one predictor to assign observations to classes. However, the strength of these classifiers was that they tended to select a different subset of variables each time the training data were reweighted. While Breiman has previously noted that such instability is a requirement for productive boosting, Friedman et al. have perhaps provided the best statistical explanation of this requirement. They have shown that boosting is simply a version of additive modeling—modeling in which several (i.e., rather than one) different multivariate models are summed to represent a decision boundary. By using several models that differ from one another, one can more accurately reproduce the
nature of complex decision boundaries. My research supported Friedman’s perspective because a single, complex tree containing up to 20 terminal nodes always had lower predictive accuracy than the best ensemble of simpler classifiers.

In contrast, boosting was unproductive with the one stable classifier tested, logistic regression. In none of the three scenarios examined did boosting improve the performance of LR. This finding was true even under conditions where LR acted as a weak classifier (i.e., had low predictive accuracy) as in Scenario 2 ($\mu_1 = 0.1$). In such cases, LR did select only a subset of the 10 predictor variables through which to classify the training data at the first boosting step. However, LR failed to identify a different subset of variables by which to make predictions once the data had been reweighted. Instead, the coefficients of all the predictors were decreased substantially, making the second classifier’s contribution to class assignments negligible. Thus, an ensemble of LR classifiers actually incorporated only a single LR classifier of any significance. Breiman and others have noted that boosting was unproductive with other stable classifiers, such as LDA and nearest-neighbor approaches, and this research was consistent with those previous findings.

Second, boosting was most effective when the complexity of the base classifier matched the complexity of the decision boundary being modeled. For example, stumps, which are capable of modeling only main effects, were the best classifiers when used with the linear decision boundary of Scenario 1. In contrast, two-level trees, which are capable of modeling two-way interactions between predictors, yielded the best results when the decision boundary was quadratic, as in Scenarios 2 and 3. The importance of choosing a classifier that was matched to the decision boundary was particularly important when the training data set contained less information, such as when there were fewer training observations or when there was more
overlap between the distributions of cases and controls (e.g., see Figure 8). When the training sets were information rich, this matching of complexity between boundary and classifier became less important. Often, the predictive accuracy of stumps and trees were very similar when the training set contained more information about the boundary between classes.

Friedman et al. noted the influence of this correspondence of complexity between base classifiers and decision boundaries in their research. They also stated that most natural decision boundaries rarely include interactions between terms that are higher than second order. Thus, for most cases, two-level trees are the most complex base classifiers that are required to model a decision boundary. Because the decision boundaries used in this research contained up to only second-order interactions, trees having three or more levels were not investigated with boosting.

**Comparison of Boosted and Stable Classification Methods**

The primary goal of this research was to compare the predictive accuracy of the best boosted classifiers with that of LDA/QDA and LR. To achieve this end, three simulation scenarios were selected that were intended to favor one or more of the methods over the others. Scenario 1 used a linear decision boundary that was intended to provide LDA and LR with a predictive advantage over boosted trees. Scenario 2 used a quadratic decision boundary intended to give QDA a predictive advantage over both LR and boosted trees. Scenario 3 also involved a quadratic decision boundary, but the populations of cases and controls were not normally distributed. This last scenario was intended to give LR and boosted trees an advantage over QDA.

Discriminant analysis (LDA/QDA) was found to have a higher predictive accuracy than boosted classification trees for all three scenarios. Superior performance was expected for LDA/QDA in Scenarios 1 and 2 where the underlying assumptions of discriminant analysis were
met. However, even when the underlying assumption of normality was violated, QDA gave lower test error rates than boosted trees (whether the classifiers were stumps or two-level trees). While the precise reasons for this superior performance in the face of a violated assumption are unclear, other studies have shown that LDA and QDA both work well with a number of different types of data sets. Hastie et al. have suggested that the robustness of discriminant analysis may be due to the method’s ability to model decision boundaries with low variance. Thus, even though discriminant analysis may create biased estimates of decision boundaries when the method’s distributional assumptions are violated, these estimates may still be acceptable due to their low variance.

In contrast, the performance of LR relative to boosted trees was dependent upon the scenario investigated. As expected, LR offered superior performance to boosted trees when the decision boundary was a linear function of the vector of predictor variables (Scenario 1). In this linear model, both LR and LDA created class assignments by using the log odds of the posterior class probabilities given \( X \), \( \log \left( \frac{P(Y=1|X)}{P(Y=0|X)} \right) \). Thus, it is no surprise that LR, like LDA, yielded test error rates that were very close to Fisher’s minimum misclassification rate here.

The ability of LR to yield optimal combinations of predictors depended upon the ability of the logistic model to reflect the true distributions of the underlying populations. Consequently, the predictive accuracy of LR was highly dependent on proper specification of the logistic model. When the logistic model was properly specified in Scenario 3, LR had not only the lowest error rate of all five classification methods, but it also had an error rate that closely approached the Bayes error rate. This result was expected given that the properly specified model used the same function of the predictor variables that was initially used to deterministically assign observations to classes.
However, when even 2 out of the 10 predictor variables were improperly specified in Scenario 3, the test error rate for LR dropped below that of boosted trees (large training set). This disparity was less pronounced when the data set was medium-sized; in this case, half of the predictor variables could be improperly specified before LR’s test error rate fell below that of boosted two-level trees. Consequently, it seems that boosting should be preferred in situations where a large amount of training data is available and the researcher is uncertain of the underlying statistical model or disease model. In contrast, if a physically sensible model can be postulated for the data, it may yield better results than boosting, especially when the amount of data available is more modest.

**Study Limitations**

The primary limitation of this study was that it was conducted solely within the classical regime, where the number of observations in the training data set was greater than the number of predictor variables. This research used a minimum of 10 training observations per predictor variable per class. It is possible that the use of smaller training sets (e.g., those having 2 training observations per predictor per class) may have led to different results. It is likely that the findings of this research do not apply to regimes where the number of predictor variables is larger than the number of available training observations (e.g., as in microarray analysis). Under such conditions, the estimates of required model parameters (e.g., covariance matrices in discriminant analysis) are likely to be unstable or indeterminate, leading to poor predictive performance for these two methods. This view is supported by the recent work of Dudoit et al., who showed that LDA had lower predictive accuracy for the classification of tumors using gene expression data.35
Other types of perturbations of the underlying populations or of the training data may have also led to different results. For example, Efron has suggested that LDA performs well even when the underlying populations are not normally distributed, as long as their distributions fall within the exponential family. However, had the distributions of cases and controls been selected to fall into another distributional family, LDA/QDA may have shown poorer performance. Likewise, training sets that contain several outliers may bias parameter estimates in both discriminant analysis and LR. In such cases, the use of boosted classification trees, which are less sensitive to outliers, may have produced better results with such data. Another departure from the current scenarios would have been to add noise to the classification data. Dietterich has shown that the performance of boosting deteriorates substantially in the presence of classification noise. Perhaps LDA/QDA and LR would have had advantages over boosted classification trees when using noisy data. Dietterich also suggests that bagging may be more helpful than boosting for noisy data.

This research used only one type of base classifier, classification trees, because these are the classifiers most commonly used with boosting. However, other types of base classifiers have been used for boosting, such as neural nets. It is unclear whether the results of the current study would have held if another type of base classifier would have been chosen, such as neural nets. Additionally, the correlations of the predictor variables in Scenario 2 were chosen to be weak to moderate. It is possible that multicollinearity would have been more of an issue for LR had these correlations been stronger or had involved a larger number of variables. Such a choice could have improved the predictive accuracy of boosted classification trees relative to LR.

Finally, because this research represented an artificial problem, the overall misclassification rate for the test set was used to compare the performance of the various
classification methods. This choice meant that sensitivity and specificity were deemed of equal importance. However, in the development of real-world diagnostic tests, sensitivity and specificity often differ in importance.\textsuperscript{42} For example, in diagnostic tests that are intended to be used for widespread screening, specificity is usually kept high to avoid having too many false positive results. Typically, one would use a Receiver Operating Characteristic (ROC) curve or the specificity at a fixed sensitivity to compare the performance of different diagnostic tests.\textsuperscript{42,51}

Because this research used only the overall test error rate to measure performance, it provided no insight into how the various classification methods might compare at points far from the center of the ROC curve.

**Future Research**

When distributional assumptions were met and models were properly specified, this research found that the stable classification methods of LR and LDA/QDA had superior predictive accuracy to boosted classification trees. In part, this may reflect the fact that boosted classifiers are a series of piece-wise approximations to a decision boundary rather than a precise analytic solution to that boundary. These results raise the question of whether ways can be found to improve the predictive accuracy of boosted classifiers.

As Friedman et al. have noted, the diversity of the individual classifiers in an ensemble is important to the success of boosting.\textsuperscript{26} Consequently, one way to improve the performance of boosted classifiers is to intentionally increase the diversity of the classifiers in the ensemble. Boosting tries to achieve this goal by constructing each classifier on a different, independent version of the training data. However, other methods have recently been suggested for improving
the diversity of classifiers, such as combining classifiers of different types (e.g., neural nets and classification trees).\textsuperscript{52-53} In developing a screening test for osteoporosis, Wang et al. have shown that a hybrid classifier that contained both classification trees and neural nets had greater predictive accuracy than a “pure” ensemble of classifiers that contained only trees or only neural nets.\textsuperscript{54} Likewise, Bonissone et al. showed that hybrid classifiers using various combinations of trees, neural nets, multivariate splines, and support vector machines improved the predictive accuracy of a decision engine for underwriting insurance applications.\textsuperscript{55} Thus, an interesting experiment would be to subject the current data to a boosting procedure in which the reweighted data were alternately presented to classification tree and neural net classifiers. The error rate of the hybrid ensemble might more closely approach the Bayes error rates than that of a pure ensemble of trees.

Other researchers have tried to devise measures of diversity for classifiers and to use these measures to select diverse classifiers for the ensemble. Such measures have included the Q statistic,\textsuperscript{56} the entropy measure,\textsuperscript{57} and the coincident failure diversity measure,\textsuperscript{58} among others. However, Narasimhamurthy has recently concluded that many measures of diversity are context sensitive, making it difficult from a practical viewpoint to use such measures to select classifiers.\textsuperscript{59}

Another key finding of this research was that boosting failed to improve the predictive accuracy of LR. This finding was consistent with previous research that showed that boosting was unproductive when used with stable classification methods.\textsuperscript{30-31,33} This raised the question of whether one can destabilize a stable classifier in such a way that boosting becomes a productive process. This question was recently addressed by Lu et al. who developed a face recognition method that combined boosting and discriminant analysis.\textsuperscript{60} Lu et al. suggested that stable
classifiers can be destabilized by exposing each stable classifier to a very limited portion of the training data set. By doing so, they found that stable learners became unstable and that boosting was productive with these classifiers. Their boosted discriminant classifiers cut misclassification rates by approximately 50% compared to single LDA classifiers.

Thus, an interesting extension of the present research would be to increase the dimensionality of the boundary problem and to see whether boosting would work with either discriminant analysis or LR. To help define the conditions under which LDA classifiers would become unstable, Lu et al. introduced a measure called the learning difficulty degree (LLD). They defined LLD as follows: \( \text{LLD} = \frac{\text{# training examples per subject}}{\text{total number of subjects}} \). Small values of LLD correspond to unstable classifiers, large values to stable classifiers. For their face recognition application, Lu et al. found that values of LLD in the range of 0.03 - 0.07 produced LDA classifiers that were productive with boosting. In contrast, the LLDs used in my research were 500 and 50 for the large and medium-sized data sets, respectively. This was far outside the range where LDA or LR might be useful classifiers for boosting. By increasing the dimensionality of the problem, I might be able to bring the LDA/LR classifiers within the optimal range.

**Conclusions**

In summary, this work led to the following conclusions:

1. Consistent with previous research, boosting was found to be productive with unstable classifiers but unproductive with stable classifiers (i.e., LR).
2. Boosting performed best when the complexity of the base classifier matched the complexity of the decision boundary being modeled. In practice, both stumps and two-level trees should be tried as classifiers in combination with a cross-validation method to see which classifier produces the best results.

3. Stable classification methods outperformed boosted classifiers when the underlying assumptions for the stable classification methods were met.

4. When underlying method assumptions were violated (i.e., distributions were non-normal for LDA/QDA or the model was misspecified for LR), discriminant analysis appeared to be the more robust method of the stable classification methods.

5. Boosting may be favored in situations where there is much training data and the underlying model of disease is poorly understood. The method performs nearly as well as discriminant analysis and will likely be superior to LR in such cases.
Bibliography


Appendix 1. Real AdaBoost Program

## Initialization and Definitions

```r
library(rpart)
library(MASS)
nruns <- 130
mfinal <- 200
store_stump <- matrix(0, nruns, mfinal)
store_tree <- matrix(0, nruns, mfinal)
store_best <- matrix(0, nruns)
store_log_reg <- matrix(0, nruns, mfinal)
store_lda <- matrix(0, nruns)
for (k in 1:nruns)
{
    # Insert the desired data generation module here!!!

    ### PERFORM BOOSTING using STUMPS

    cntrl <- rpart.control(maxdepth=1, minsplit=learn-1, minbucket = 1,
                           maxsurrogate = 0, usesurrogate = 0, maxcompete = 0,
                           cp = 0, xval = 0)

    Flearn <- numeric(learn)
    Ftest <- numeric(test)
    ptest <- matrix(0,test,mfinal)
    w <- rep(1/learn,learn)
    error <- matrix(0, mfinal,2)
    error[,1] <- 1:mfinal
    names <- c("step", "error")
    colnames(error) <- names
    blearn <- matrix(0,learn)
    btest <- matrix(0,test)

    ### Shifting the labels and define logit function

    ylearn[ylearn==0] <- -1
    ytest[ytest==0] <- -1
    logit <- function(x) log(x/(1-x))

    ### Perform the boosting loop using stumps

    for (m in 1:mfinal)
    {
        bx <- xlearn
        fit <- rpart(ylearn ~ bx, weights = w/mean(w), control = cntrl, method="class")
        bx <- xtest
        blearn <- predict(fit)[,2]
        btest <- predict(fit, newdata = data.frame(bx))[ ,2]

        flearn <- pmin(0.5*logit(blearn), 1e+8)
        ftest <- pmin(0.5*logit(btest), 1e+8)
    }
```

```
Appendix 1. Real AdaBoost Program (continued)

\[
\begin{align*}
F_{\text{learn}} &\leftarrow F_{\text{learn}} + f_{\text{learn}} \\
F_{\text{test}} &\leftarrow F_{\text{test}} + f_{\text{test}} \\
sign_{\text{learn}} &\leftarrow \text{sign}(F_{\text{learn}}) \\
sign_{\text{test}} &\leftarrow \text{sign}(F_{\text{test}}) \\
\text{err}_{\text{test}} &\leftarrow \frac{\text{sum}(\text{abs}(\text{sign}_{\text{test}} - y_{\text{test}}))/(2*\text{test})}{\text{err}_{\text{test}}} \\
\text{err}_{\text{learn}} &\leftarrow \frac{\text{sum}(\text{abs}(\text{sign}_{\text{learn}} - y_{\text{learn}}))/(2*\text{learn})}{\text{error}[m,2]} \\
w &\leftarrow w*\text{exp}(-y_{\text{learn}}*f_{\text{learn}}) \\
w &\leftarrow \text{pmax}(w/\text{sum}(w), 1e-24) \\
\}
\text{store}_{\text{stump}}[k, ] &\leftarrow \text{error}[ ,2]
\end{align*}
\]

############################################################
##### Insert the desired data generation module here!!!
############################################################

###### PERFORM BOOSTING using TWO-LEVEL TREES
################################################
\text{cntrl} &\leftarrow \text{rpart.control(maxdepth=2, minsplit=50, #minbucket = 1,} \\
& \text{maxsurrogate = 0, usesurrogate = 0, maxcompete = 0,} \\
& \text{cp = 0, xval = 0})
\text{F}_{\text{learn}} &\leftarrow \text{numeric(learn)} \\
\text{F}_{\text{test}} &\leftarrow \text{numeric(test)} \\
\text{ptest} &\leftarrow \text{matrix(0,test,mfinal)} \\
w &\leftarrow \text{rep}(1/\text{learn}, \text{learn}) \\
\text{error} &\leftarrow \text{matrix(0, mfinal,2)} \\
\text{error}[1] &\leftarrow \text{1:mfinal} \\
\text{names} &\leftarrow \text{c("step", "error")} \\
\text{colnames(error)} &\leftarrow \text{names} \\
\text{blearn} &\leftarrow \text{matrix(0,learn)} \\
\text{btest} &\leftarrow \text{matrix(0,test)} \\
## Shifting the labels and define logit function \\
\text{ylearn}[ylearn==0] &\leftarrow -1 \\
\text{ytest}[ytest==0] &\leftarrow -1 \\
\text{logit} &\leftarrow \text{function(x) log(x/(1-x))} \\
## Perform the boosting loop using stumps \\
\text{for (m in 1:mfinal)}
\{
\text{bx} &\leftarrow \text{xlearn} \\
\text{fit} &\leftarrow \text{rpart(ylearn ~ \text{bx}, weights = w/mean(w), control = \text{cntrl}, method="class")} \\
\text{bx} &\leftarrow \text{xtest} \\
\text{blearn} &\leftarrow \text{predict(fit)[,2]} \\
\text{btest} &\leftarrow \text{predict(fit, newdata = \text{data.frame(bx)})[,2]} \\
\text{flearn} &\leftarrow \text{pmin}(0.5*\text{logit(blearn)}, 1e+8) \\
\text{ftest} &\leftarrow \text{pmin}(0.5*\text{logit(btest)}, 1e+8)
\}
\]
Appendix 1. Real AdaBoost Program (continued)

Flearn <- Flearn + flearn
Ftest <- Ftest + ftest
sign_learn <- sign(Flearn)
sign_test <- sign(Ftest)

err_test <- sum(abs(sign_test - ytest))/(2*test)
err_learn <- sum(abs(sign_learn - ylearn))/(2*learn)
error[m,2] <- err_test

w <- w*exp(-ylearn*flearn)
w <- pmax(w/sum(w), 1e-24)
}
store_tree[k,] <- error[,2]

#############################################################################################
##### Insert the desired data generation module here!!!
#############################################################################################

##### Find BEST MULTI-LEVEL TREE #######
#################################################################
ylearn[ylearn==0] <- -1
ytest[ytest==0] <- -1

cntrl <- rpart.control(maxdepth=30, minsplit=10, #minbucket = 1,
maxsurrogate = 0, usesurrogate = 0, maxcompete = 0,
 cp = 0.005, xval = 0)

bx <- xlearn
fit <- rpart(ylearn ~ bx, control = cntrl, method="class")
bx <- xtest
blearn <- predict(fit)[,2]
btest <- predict(fit, newdata = data.frame(bx))[,2]

sign_test <- sign(btest-0.5)
err_test <- sum(abs(sign_test - ytest))/(2*test)
store_best[k] <- err_test

#############################################################################################
##### Insert the desired data generation module here!!!
#############################################################################################

##### USE LOGISTIC REGRESSION MODEL AS A BASE CLASSIFIER
#############################################################

ylearn[ylearn== -1] <- 0
ytest[ytest== -1] <- 0

Flearn <- numeric(learn)
Ftest <- numeric(test)

# w <- rep(1,learn)
w <- rep(1/learn,learn)
Appendix 1. Real AdaBoost Program (continued)

```r
error <- matrix(0, mfinal,2)
error[,1] <- 1:mfinal
names <- c("step", "error")
colnames(error) <- names
blearn <- matrix(0,learn)
btest <- matrix(0,test)

for (m in 1:mfinal)
{
  bx <- xlearn
glm.out <- glm(ylearn ~ bx, weights = w/mean(w), binomial)
  if (m <= 3 && k==1) print(glm.out)

  bx <- xtest
  blearn <- predict(glm.out)
btest <- predict(glm.out, newdata = data.frame(bx))

  flearn <- pmin(0.5*blearn, 1e+8)
  ftest <- pmin(0.5*btest, 1e+8)

  ylearn[ylearn==0] <- -1
  ytest[ytest==0] <- -1

  Flearn <- Flearn + flearn
  Ftest <- Ftest + ftest
  sign_learn <- sign(Flearn)
  sign_test <- sign(Ftest)

  err_test <- sum(abs(sign_test - ytest))/(2*test)
  err_learn <- sum(abs(sign_learn - ylearn))/(2*learn)
  error[m,2] <- err_test

  w <- w*exp(-ylearn*flearn)
  w <- pmax(w/sum(w), 1e-24)

  ylearn[ylearn==-1] <- 0
  ytest[ytest==-1] <- 0
}

store_log_reg[k, ] <- error[,2]
```

########################################################################
##### Insert the desired data generation module here!!!
########################################################################

##### Apply Linear Discriminant Analysis
########################################################################

```r
fit <- lda(x = xlearn, grouping = ylearn)
pred_test <- predict(fit, newdata=xtest)

a <- as.numeric(pred_test$class)
```
Appendix 1. Real AdaBoost Program (continued)

a[α==1]  <- 0
a[α==2]  <- 1
delta  <- abs(ytest - a)
test_err  <- sum(delta)/test
store_lda[k]  <- test_err

### CALCULATE AND PRINT SUMMARY DATA

stump_mean  <- matrix(0,1,mfinal)
stump_sd  <- matrix(0,1, mfinal)
stump_mean  <- apply(store_stump, 2, mean)
stump_sd  <- apply(store_stump, 2, sd)
step  <- matrix(1:mfinal,1,mfinal)
order_st_mean  <- order(stump_mean)
min_stump_step  <- step[order_st_mean][1]
min_stump_sd  <- stump_sd[order_st_mean][1]

err_start  <- c("stump mcr at step 1:","round(stump_mean[1],4),"+-",round(stump_sd[1],5))
err_start
err_min  <- c("stump minimum mcr:","round(min(stump_mean),4),"+-",round(min_stump_sd,5),"at step",min_stump_step)
err_min
err_finish  <- c("stump mcr at last step:","round(stump_mean[mfinal],4),"+-",round(stump_sd[mfinal],5))
err_finish

tree_mean  <- matrix(0,1,mfinal)
tree_sd  <- matrix(0,1, mfinal)
tree_mean  <- apply(store_tree, 2, mean)
tree_sd  <- apply(store_tree, 2, sd)
order_tr_mean  <- order(tree_mean)
min_tree_step  <- step[order_tr_mean][1]
min_tree_sd  <- tree_sd[order_tr_mean][1]

err_start  <- c("tree mcr at step 1:","round(tree_mean[1],4),"+-",round(tree_sd[1],5))
err_start
err_min  <- c("tree minimum mcr:","round(min(tree_mean),4),"+-",round(min_tree_sd,5),"at step",min_tree_step)
err_min
err_finish  <- c("tree mcr at last step:","round(tree_mean[mfinal],4),"+-",round(tree_sd[mfinal],5))
err_finish

best_mean  <- c("best complex tree mean","round(mean(store_best), 4),"+-",round(sd(store_best),4))
best_mean

log_reg_mean  <- matrix(0,1,mfinal)
log_reg_sd  <- matrix(0,1, mfinal)
log_reg_mean  <- apply(store_log_reg, 2, mean)
log_reg_sd  <- apply(store_log_reg, 2, sd)
Appendix 1. Real AdaBoost Program (continued)

```r
step <- matrix(1:mfinal,1,mfinal)
order_lr_mean <- order(log_reg_mean)
min_lr_step <- step[order_lr_mean][1]
min_lr_sd <- stump_sd[order_lr_mean][1]

err_start <- c("log_reg mcr at step 1:" , round(log_reg_mean[1], 4), "+/-" , round(log_reg_sd[1], 5))
err_start
err_min <- c( "log_reg minimum mcr:" , round(min(log_reg_mean), 4), "+/-", round(min_lr_sd, 5), "at step", min_lr_step)
err_min
err_finish <- c("log_reg mcr at last step:" , round(log_reg_mean[mfinal], 4), "+/-" ,
 round(log_reg_sd[mfinal], 5))
err_finish

lda_mean <- c("lda mean", round(mean(store_lda), 4), "+/-" , round(sd(store_lda), 5))
lda_mean

mu_del <- mu1 - mu2
alpha <- t(mu_del)%*%ginv(sigma1)%*%mu_del
lda_min <- round(pnorm(-0.5*sqrt(alpha)), 4)
lda_min
alpha

steps <- matrix(1:mfinal, nrow=mfinal, ncol=1, byrow=TRUE)
combo <- as.matrix(cbind(stump_mean, tree_mean, log_reg_mean))
matplot(steps, combo, type="l", xlab="Boosting Steps" , ylab="Test Error Rate", col=c(1,1,1), lty=c(1,2,4))

abline(mean(store_best),0, lty=5)
abline(mean(store_lda), 0, lty=6)

text(150 ,0.3 ,"stumps")
text(180,0.286 ,"two-level trees")
text(160,0.359 ,"single six-level tree")
text(80, 0.235,"logistic regression")
text(10 ,0.235 ,"LDA")
```
Appendix 2. R Program for the Estimation of the Bayes Error Rate

### Program for calculation of Bayes Error rates for Scenario #2

```r
library(MASS); library(mvtnorm)

n.est  <- 10
mu1   <- rep(0,10)
mu2   <- rep(0.1, 10)
incr  <- rep(0.4, 10)
sigma1 <- diag(rep(1,10),10)
s1    <- c(1, 0.3, 0.3, rep(0,7))
s2    <- c(0.3, 1, 0.3, rep(0,7))
s3    <- c(0.3, 0.3, 1, rep(0,7))
s4    <- c(rep(0.3), 1, 0.5, 0.5, rep(0.4))
s5    <- c(rep(0.3), 0.5, 1, 0.5, rep(0.4))
s6    <- c(rep(0.3), 0.5, 0.5, 1, rep(0.4))
s7    <- c(rep(0.6), 1, rep(0.3))
s8    <- c(rep(0.7), 1, rep(0.2))
s9    <- c(rep(0.8), 1, 0)
s10   <- c(rep(0.9), 1)
sigma2 <- rbind(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10)
sigma2 <- matrix(0, 3, n.est, byrow=T)

for (k in 1:3)
{
error  <- numeric(n.est)
for (m in 1:n.est)
{
  xctrl  <- mvrnorm(10000, mu1, sigma1)
  xcase  <- mvrnorm(10000, mu2, sigma2)
  xlearn <- rbind(xctrl,xcase)
  ylearn <- c(rep(0,10000), rep(1,10000))
  learn  <- dim(xlearn)[1]
  pdens1 <- numeric(learn)
  pdens2 <- numeric(learn)
  for (i in 1:learn) pdens1[i] <- dmvnorm(xlearn[i,], mu1, sigma1)
  for (i in 1:learn) pdens2[i] <- dmvnorm(xlearn[i,], mu2, sigma2)
  delta  <- numeric(learn); delta <- pdens2 - pdens1
  sign   <- sign(delta)
  ylearn[ylearn==0]  <- -1
  error[m]  <- sum(abs(ylearn-sign))/(2*learn)
}
summary[k, ]  <- error
mu2    <- mu2 + incr
}

summary
mean_summary  <- apply(summary,1,mean); mean_summary
sd_summary   <- apply(summary, 1, sd); sd_summary
```

83
Appendix 3. R Program for Generating the Training and Test Data Sets

## SCENARIO #1 - INDEPENDENT PREDICTORS: NORMAL DISTRIBUTION (N=2000)
## Controls have Y ~ N_{10}(0, I)
## Cases have Y ~ N_{10}(mu2, I)

```r
mu1 <- rep(0,10);  mu2 <- rep(0.75,10)
sigma1 <- diag(rep(1,10),10);  sigma2 <- diag(rep(1,10),10)
xctrl <- mvrnorm(1000, mu1, sigma1)
xcase <- mvrnorm(1000, mu2, sigma2)
xlearn <- rbind(xctrl, xcase)
ylearn <- c(rep(0,1000), rep(1,1000))
xctrl1 <- mvrnorm(1000, mu1, sigma1)
xcase1 <- mvrnorm(1000, mu2, sigma2)
xtest <- rbind(xctrl1, xcase1)
ytest <- c(rep(0,1000), rep(1,1000))
learn <- dim(xlearn)[1];   test <- dim(xtest)[1]
```

## SCENARIO #2 - CORRELATED PREDICTORS: NORMAL DISTRIBUTION (N=2000)
## Controls have Y ~ N_{10}(0,I); Cases have Y ~ N_{10}(mu2, sigma2)

```r
mu1 <- rep(0,10);  mu2 <- rep(0.1,10)
sigma1 <- diag(rep(1,10),10)
s1 <- c(1, 0.3, 0.3, rep(0,7))
s2 <- c(0.3, 1, 0.3, rep(0,7))
s3 <- c(0.3, 0.3, 1, rep(0,7))
s4 <- c(rep(0.3), 1, 0.5, 0.5, rep(0,4))
s5 <- c(rep(0.3), 0.5, 1, 0.5, rep(0,4))
s6 <- c(rep(0.3), 0.5, 0.5, 1, rep(0,4))
s7 <- c(rep(0.6), 1, rep(0,3))
s8 <- c(rep(0.7), 1, rep(0,2))
s9 <- c(rep(0.8), 1, 0)
s10 <- c(rep(0.9), 1)
sigma2 <- rbind(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10)
sigma2 <- as.matrix(sigma2)
xctrl <- mvrnorm(1000, mu1, sigma1)
xcase <- mvrnorm(1000, mu2, sigma2)
xlearn <- rbind(xctrl, xcase)
ylearn <- c(rep(0,1000), rep(1,1000))
xctrl1 <- mvrnorm(1000, mu1, sigma1)
xcase1 <- mvrnorm(1000, mu2, sigma2)
xtest <- rbind(xctrl1, xcase1)
ytest <- c(rep(0,1000), rep(1,1000))
learn <- dim(xlearn)[1];   test <- dim(xtest)[1]
```
Appendix 3. R Program for Generating the Training and Test Data Sets (continued)

******************************************************************************
## SCENARIO #3 - FRIEDMAN'S NESTED SPHERES
## Each observation is initially generated as Y ~ N(0, 1)
## Y = 1 when sum (X_i**2) > 9.34 (i.e., > chi-square(10), alpha = 0.5); otherwise, Y = 0

```r
xlearn <- matrix(rnorm(20000), 2000, 10, byrow=T)
dimnames(xlearn) <- NULL
square_x <- xlearn*xlearn
chi_sq <- apply(square_x,1,sum)
chi_sq[chi_sq <= 9.34] <- 0
chi_sq[chi_sq > 9.34] <- 1
ylearn <- chi_sq

xtest <- matrix(rnorm(20000), 2000, 10, byrow=T)
dimnames(xtest) <- NULL
square <- xtest*xtest
chi <- apply(square,1,sum)
chi[chi <= 9.34] <- 0
chi[chi > 9.34] <- 1
ytest <- chi

learn <- dim(xlearn)[1]
test <- dim(xtest)[1]
```

#Note: Two R libraries, MASS and mvtnorm, must be called in order for these data generation modules to work. These libraries are called within the Real AdaBoost program in which these modules are to be inserted.