View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Minnesota Digital Conservancy

Variable Selection and Prediction in "Messy" High-Dimensional Data

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Benjamin Timothy Brown

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Julian Wolfson

July, 2017

© Benjamin Timothy Brown 2017
ALL RIGHTS RESERVED

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school. My advisor Julian Wolfson helped get me here when I thought I might not make it. Sally Olander had the answer to every question I had along the way. Brad Carlin kept me on track and provided a great deal of needed support.

Dedication

To my wife, Lindsey, for encouraging me for the past three and a half years and supporting me through the many long nights of research.

Abstract

When dealing with high-dimensional data, performing variable selection in a regression model reduces statistical noise and simplifies interpretation. There are many ways to perform variable selection when standard regression assumptions are met, but few that work well when one or more assumptions is violated. In this thesis, we propose three variable selection methods that outperform existing methods in such “messy data” situations where standard regression assumptions are violated. First, we introduce Thresholded EEBoost (ThrEEBoost), an iterative algorithm which applies a gradient boosting type algorithm to estimating equations. Extending its progenitor, EEBoost (Wolfson, 2011), ThrEEBoost allows multiple coefficients to be updated at each iteration. The number of coefficients updated is controlled by a threshold parameter on the magnitude of the estimating equation. By allowing more coefficients to be updated at each iteration, ThrEEBoost can explore a greater diversity of variable selection “paths” (i.e., sequences of coefficient vectors) through the model space, possibly finding models with smaller prediction error than any of those on the path defined by EEBoost. In a simulation of data with correlated outcomes, ThrEEBoost reduced prediction error compared to more naive methods and the less flexible EEBoost. We also applied our method to the Box Lunch Study where we found that we were able to reduce our error in predicting BMI from longitudinal data. Next, we propose a novel method, MEBoost, for variable selection and prediction when covariates are measured with error. To do this, we incorporate a measurement error corrected score function due to Nakamura (1990) into the ThrEEBoost framework. In both simulated and real data, MEBoost outperformed the CoCoLasso (Datta and Zou, 2017), a recently proposed penalization-based approach to variable selection in the presence of measurement error, and the (non-measurement error corrected) Lasso. Lastly, we consider the case where multiple regression assumptions may be simultaneously violated. Motivated by the idea of stacking, specifically the SuperLearner technique (Van Der Laan et al., 2007), we propose a novel method, Super Learner Estimating Equation Boosting (SuperBoost). SuperBoost performs variable selection in the presence of multiple data challenges by combining the results from variable selection procedures which are each tailored to address a different regression assumption violation. The ThrEEBoost framework is a natural fit for this approach, since the component "learners" (i.e., violation-specific variable selection techniques) are fairly straightforward to construct and implement by using various estimating equations. We illustrate the application of SuperBoost on simulated data with both correlated outcomes and covariate measurement error, and show that it performs as well or better than methods which address only one (or neither) of these factors.

Contents

Acknowledgements **i**

Dedication **ii**

Abstract **iii**

List of Tables **viii**

List of Figures **x**

1 Introduction **1**

2 ThrEEBoost **4**

2.1 Introduction 4

2.2 Boosting, EEBoost, and ThrEEBoost 6

2.2.1 EEBoost 8

2.2.2 Diversifying variable selection paths 9

2.2.3 ThrEEBoost: Thresholded EEBoost 10

2.2.4 Selecting the best model 11

2.3 Simulation Study 13

2.3.1 Sparse regression model with correlated outcomes 13

2.3.2 Less sparse regression model with correlated outcomes 18

2.4 Data application - Box Lunch Study 22

2.5 Discussion 24
2.6 Supplementary Materials 26

3 MEBoost **27**

3.1 Introduction 27
3.2 Background 29
3.2.1 Regression in the Presence of Covariate Measurement Error . . . 29
3.2.2 Variable selection in the Presence of Measurement Error 29
3.2.3 Lasso in the Presence of Measurement Error 30
3.2.4 The Convex Conditioned Lasso (CoCoLasso) 32
3.3 MEBoost: Measurement Error Boosting 33
3.3.1 Corrected Score Function 34
3.3.2 The MEBoost Algorithm 35
3.4 Simulation Study 38
3.4.1 Simulation Set-up 38
3.4.2 Simulation Results 41
3.5 Data Application 43
3.6 Discussion 45
3.7 Tables and Figures 48

4 SuperBoost **52**

4.1 Introduction 52
4.2 Literature Review 54
4.2.1 Variable selection when regression assumptions are violated . . . 54
4.2.2 Stacking 55
4.2.3 Super Learner 57
4.3 Methods 59

4.3.1 Boosting 59
4.3.2 Combining the methods 61
4.3.3 SuperBoost 62
4.4 Simulation 65
4.4.1 Set-up 65
4.4.2 Results 66
4.5 Discussion 70

5 Conclusion and Discussion **76**

References **78**

List of Tables

2.1 Mean minimum prediction error (a), median variable selection (b) sensitivity and (c) specificity, (d) mean number of iterations (25th and 75th percentile) to attain minimum prediction error, (e) minimum mean QIC, and (f) proportion of simulations where algorithm did not find a unique minimum MSE for ThrEEBoost in the sparse true model under different values of the threshold, τ and correlation between intra-individual observations, ρ. Results are based on 1000 simulations, each with 500 iterations. 16

2.2 Mean minimum prediction error (a), median variable selection (b) sensitivity and (c) specificity, (d) mean number of iterations (25th and 75th percentile) to attain minimum prediction error, (e) minimum mean QIC, and (f) proportion of simulations where algorithm did not find a unique minimum MSE for ThrEEBoost in the less sparse true model under different values of the threshold, τ, and correlation between intra-individual observations, ρ. Results are based on 1000 simulations, each with 1500 ThrEEBoost iterations. 19

2.3 Coefficients for the optimal ThrEEBoost ($\tau = 0.4$) and LASSO models selected by cross-validated MSE. Small coefficients (magnitude < 0.05) are omitted. "–" indicates that the variable was not selected in the model. 22

2.4 Estimated mean squared prediction error for ThrEEBoost and LASSO models. (CV MSE) denotes models selected by minimizing cross-validated MSE. 23

3.1 Performance metrics for the 1,000 simulations in various measurement error scenarios. The models were selected at the point with minimum MSE-M. 50

3.2 Coefficients, Deviance, and MSE-M from selected models for MEBoost with specified value of τ and $\hat{\delta}^2_D$ and the Lasso. Small coefficients (magnitude < 0.05) are omitted. "-" indicates that the variable was not selected in the model. 51

4.1 List of parameter values for each scenario. n is the number of clusters, σ^2 is the residual variance, δ is the standard deviation of the measurement error, ρ is the pairwise correlation in outcomes within clusters, β_m is the effect size of mismeasured predictors, β_g is the effect size of predictors measured without error. Scenario b in each setting has fewer clusters and higher residual variance, giving us less information to reach an accurate estimate. 65

4.2 Performance metrics for scenario 1a. 67

4.3 Performance metrics for scenario 1b. 68

4.4 Performance metrics for scenario 2a. 69

4.5 Performance metrics for scenario 2b. 70

4.6 Performance metrics for scenario 3a. 71

4.7 Performance metrics for scenario 3b. 72

List of Figures

2.1 Average L_1 distances from the true β (top row), estimated coefficient values (middle row) and MSPE (bottow row) across iterations for various values of τ, when data are generated from a very sparse true regression model with an intra-individual correlation of $\rho = 0.3$. The solid, dashed, and dotted lines in the coefficient plots (middle row) represent coefficients with true values of 0.5, 0.2, and 0.0 respectively. Results are based on 1000 simulations, each with 500 ThrEEBoost iterations. The solid vertical lines show the iteration where the minimum mean squared error is achieved in each scenario. 15

2.2 The distribution of selected τ values via cross-validation. For each value of ρ, the median τ_{CV} selected was 0.58. 17

2.3 Average L_1 distances from the true β (top row), estimated coefficient values (middle row) and MSPE (bottow row) across iterations for various values of τ, when data are generated from a less sparse true regression model with an intra-individual correlation of $\rho = 0.3$. The solid, dashed, and dotted lines in the coefficient plots (middle row) represent coefficients with true values of 0.5, 0.2, and 0.0 respectively. Results are based on 1000 simulations, each with 1500 ThrEEBoost iterations. The solid vertical lines show the iteration where the minimum mean squared error is achieved in each scenario. 20

2.4 The distribution of selected τ values via cross-validation. For each value of ρ, the median τ_{CV} selected were 0.38, 0.40, and 0.38. 21

2.5 Average QIC when data are generated from a less sparse true regression model with an intra-individual correlation of $\rho = 0.3$. Results are based on 1000 simulations, each with 1500 ThrEEBoost iterations. 21

2.6 Coefficient magnitudes for the optimal models (chosen by cross-validated MSE) for different values of τ. Each row corresponds to a different variable; darker shades of gray correspond to higher coefficient magnitudes. The names of the variables are displayed on the right; a data dictionary giving the variable descriptions is provided in the Supplementary Materials. 25

3.1 Summary statistics for the scenario with varying levels of independent measurement errors. Plots are of MSE, MSE-M, L_1 distance from $\hat{\beta}$ to the true value of β, sensitivity, and specificity across the mean path over 1,000 simulations. 48

3.2 In the scenario, with iid measurement error, 1,000 simulations were conducted comparing the variable selection path of MEBoost to CoCoLasso. Plots are of the number of nonzero coefficients included in the model, the portion of coefficients that are 'mismatched' with only one of the two models selecting a certain variable, and the element-wise distance in a coefficient between the two methods. 49

4.1 Model weights and traceplots for mean paths of Super Learner models in scenario 1b. 74

4.2 Performance metrics over the paths created by MEBoost in scenario 3a. 75

Chapter 1

Introduction

Driven by the ever-increasing amount of high-dimensional data in biomedicine, much recent research has focused on how to do variable selection and prediction in problems where the number of predictors, p, is large in comparison to the number of observations, n. Traditional approaches like forward selection and backward elimination are widely employed but have limitations, particularly when the number of covariates is very large. For instance, it has been shown that the first variable selected in forward selection candidate models can often be the first removed in backwards elimination (Hocking, 1976). Methods such as the Lasso (Tibshirani, 1996) and SCAD (Fan and Li, 2001) generally offer superior variable selection and predictive performance to stepwise techniques, but have been applied almost exclusively to general linear (Park et al., 2006) and survival regression models (Fan and Li, 2002).

Some authors have extended penalized approaches to more complex modeling situations such as correlated outcomes (Johnson et al., 2008), missing covariates (Yang et al., 2005), and measurement error in covariates (Datta and Zou, 2017). However, the resulting statistical procedures often involve constrained optimization of nonconvex functions, and may therefore be too computationally intensive to apply in settings where p is on the order of hundreds or thousands. Ueki (2009) proposes a smooth thresholding approach to penalizing estimating equations, with the selection threshold determined by an adaptive lasso type estimator. While smooth thresholding avoids convex optimization and therefore offers a computational speedup, the method still requires that a set of estimating equations be solved numerically for a large number of points on a two-dimensional grid of tuning parameters. Further, since the thresholding relies on an initial "full model" estimator, it is unclear how this technique generalizes to problems where p is large in relation to n.

As an alternative to penalization methods, Wolfson (2011) introduced EEBoost, a gradient descent-based method that can be used to perform variable selection for any regression problem where estimation of low-dimensional coefficients can be performed by solving an estimating equation. EEBoost iteratively constructs a set of models defined by coefficients using a modified steepest descent algorithm wherein the gradient of the loss function is replaced by the relevant estimating equation. The generic EEBoost algorithm is easily implemented using existing statistical software and can be applied to a wide variety of problems. For example, Wolfson (2011) applied EEBoost to generalized estimating equations (GEE) for correlated data, and inverse probability weighted estimating equations methods for time-to-event data with missing covariates, and Janes et al. (2012) applied it to doubly robust semiparametric efficient estimating equations for continuous outcome data.

We propose a novel method, Thresholded EEBoost (ThrEEBoost), an extension to EEBoost wherein multiple coefficients may be updated at each iteration; the number of coefficients updated is controlled by a threshold parameter on the magnitude of the estimating equation. By allowing more coefficients to be updated at each iteration, ThrEEBoost can explore a greater diversity of variable selection "paths" (i.e., sequences of coefficient vectors) through the model space, possibly finding models with smaller prediction error than any of those on the path defined by EEBoost. To evaluate our method, we will conduct a simulation with a GEE model, we will apply the method to the longitudinal data set of the Box Lunch study.

Next, we will propose an extension of our method to perform variable selection and prediction when covariates are measured with error. To do this, we utilize ThrEEBoost along with a correct score function for covariates with measurement error from Nakamura (1990). In this section, we will define the new method of Measurement Error Boost (MEBoost) and conduct a simulation study to compare it to the existing method of the CoCoLasso (Datta and Zou, 2017) which we will also describe.

Lastly, we consider that a single data set may encounter more than one data challenge. There has not been much research allocated to dealing with multiple regression assumptions being violated within one data set in either the low or high-dimensional case. We provide a general method to perform variable selection in the presence of multiple data challenges where we combine the machine learning principles of boosting and stacking, specifically the method Super Learner (Van Der Laan et al., 2007). We present a novel method, Super Learner Estimating Equation Boosting (SuperBoost) and examine it's performance through data simulated with correlated outcomes and covariate measurement error.

Chapter 2

ThrEEBoost

2.1 Introduction

Driven by the ever-increasing amount of high-dimensional data in biomedicine, much recent research has focused on how to do variable selection and prediction in problems where the number of predictors, p, is large in comparison to the number of observations, n. Traditional approaches like forward selection and backward elimination are widely employed but have limitations, particularly when the number of covariates is very large. For instance, it has been shown that the first variable selected in forward selection candidate models can often be the first removed in backwards elimination (Hocking, 1976). Methods such as the LASSO (Tibshirani, 1996) and SCAD (Fan and Li, 2001) generally offer superior variable selection and predictive performance to stepwise techniques, but have been applied almost exclusively to general linear (Park et al., 2006) and survival regression models (Fan and Li, 2002). Some authors have extended penalized approaches to more complex modeling situations such as correlated outcomes (Johnson et al., 2008) and missing covariates (Yang et al., 2005). However, the resulting statistical procedures often involve constrained optimization of nonconvex functions, and may therefore be too computationally intensive to apply in settings where p is on the order of hundreds or thousands. Ueki (2009) proposes a smooth thresholding approach to penalizing estimating equations, with the selection threshold determined by an adaptive LASSO type estimator. While smooth thresholding avoids convex optimization and therefore offers a computational speedup, the method still requires that a set of estimating equations be solved numerically for a large number of points on a two-dimensional grid of tuning parameters. Further, since the thresholding relies on an initial "full model" estimator, it is unclear how this technique generalizes to problems where p is large in relation to n.

As an alternative to penalization methods, Wolfson (2011) introduced EEBoost, a gradient descent-based method that can be used to perform variable selection for any regression problem where estimation of low-dimensional coefficients can be performed by solving an estimating equation. EEBoost iteratively constructs a set of models defined by coefficients using a modified steepest descent algorithm wherein the gradient of the loss function is replaced by the relevant estimating equation. The generic EEBoost algorithm is easily implemented using existing statistical software and can be applied to a wide variety of problems. Wolfson (2011) applied EEBoost to generalized esimtating equations (GEE) (Liang and Zeger, 1986) for correlated data, and inverse probability weighted estimating equations methods for time-to-event data with missing covariates, and Janes et al. (2012) applied it to doubly robust semiparametric efficient estimating equations for continuous outcome data.

In this paper, we propose Thresholded EEBoost (ThrEEBoost), an extension to EEBoost wherein multiple coefficients may be updated at each iteration; the number of coefficients updated is controlled by a threshold parameter on the magnitude of the estimating equation. By allowing more coefficients to be updated at each iteration, ThrEEBoost can explore a greater diversity of variable selection "paths" (i.e., sequences of coefficient vectors) through the model space, possibly finding models with smaller prediction error than any of those on the path defined by EEBoost.

2.2 Boosting, EEBoost, and ThrEEBoost

Suppose we observe outcome data $\mathbf{Y}_i$ and covariates $\boldsymbol{X}_i$, $i = 1, \ldots, n$ with $\boldsymbol{X}_i = \{X_{i1}, \ldots, X_{ip}\}$. We wish to predict future observations $\mathbf{Y}_{n+1}, \ldots, \mathbf{Y}_{n+K}$ that arise from the same distribution $F(\boldsymbol{X}, \mathbf{Y})$ as the observed data. One common approach to prediction is to use a regression model in which the relationship between the outcome and covariates is governed by the linear predictor $\boldsymbol{X}_i\boldsymbol{\beta}$. The goal, then, is to estimate a set of coefficients, $\hat{\boldsymbol{\beta}}$, that minimizes risk for a nonnegative loss function L: $R(\boldsymbol{\beta}) \equiv E_F[L(\boldsymbol{X}, \boldsymbol{\beta})]$, i.e., to obtain $\hat{\boldsymbol{\beta}}$ such that $R(\hat{\boldsymbol{\beta}}) \approx \min_{\boldsymbol{\beta}} R(\boldsymbol{\beta}) \equiv \boldsymbol{\beta}_0$. When p is small compared to n, estimation involves directly minimizing L with respect to $\boldsymbol{\beta}$ either analytically or numerically. In the case of least squares regression with independent scalars Y_i, parameter estimates are determined by $\hat{\boldsymbol{\beta}}_{LS} = \arg\min_{\boldsymbol{\beta}} \sum_i (Y_i - \boldsymbol{X}_i\boldsymbol{\beta})^2$. More generally, if a complete or partial log-likelihood ℓ is available, we can compute parameter estimates $\hat{\boldsymbol{\beta}}_{MLE} = \arg\min_{\beta}[-\ell(\boldsymbol{\beta}, X)]$. It is well known that when the number of covariates, p, is large in comparison to the sample size, n, using a subset of the p covariates to estimate Y_i will often lead to better prediction characteristics than estimating nonzero coefficients for the entire $\boldsymbol{\beta}$ vector (Wasserman, 2004). Hence, for large p, variable selection is an important step in computing $\hat{\boldsymbol{\beta}}$.

The most commonly used variable selection techniques are penalization methods which restrict the magnitude of $\boldsymbol{\beta}$ to discourage unimportant predictors from having non-zero coefficients. Stronger restrictions yield simpler models with fewer selected covariates, while weaker ones lead to more nonzero coefficient estimates. For example, the LASSO (Tibshirani, 1996) and ridge regression (Hoerl and Kennard, 1970) restrict the L_1 and L_2 norms of $\boldsymbol{\beta}$ respectively.

An alternative to penalized methods is boosting or functional gradient descent (Freund and Schapire, 1997; Friedman et al., 2000; Friedman, 2004), a variable selection technique that additively builds a model using subsets of the predictors. Given a loss function L, one sets $\boldsymbol{\beta} \equiv \boldsymbol{\beta}^{(0)} = \mathbf{0}$, and then iteratively "nudges" the entry in $\boldsymbol{\beta}$ corresponding to the element of the gradient which is largest in magnitude by some small amount ϵ. A small increment ϵ is chosen since the direction of steepest descent of L is only valid in a local neighborhood of $\boldsymbol{\beta}$. Algorithm 1 describes the steps in a generic "ϵ-boosting" algorithm. For linear regression with squared error loss, Algorithm 1 corresponds to the Forward Stagewise algorithm described in Efron et al. (2004), which is shown to be approximately equivalent (for large n and small ϵ) to Least Angle Regression and the LASSO. Prior to implementing the algorithm, all predictors need to be scaled and centered.

Algorithm 1 ϵ-boosting

```
procedure ε-BOOST
Set $\boldsymbol{\beta}^{(0)}$ to the zero $p$-vector $\mathbf{0}_p$.
    for $t = 0, \ldots, T$ do
        Compute the gradient of $L$ at the current estimate $\boldsymbol{\beta}^{(t)}$: $\boldsymbol{\Delta} = (\partial L(\boldsymbol{X}, \boldsymbol{\beta})/\partial \boldsymbol{\beta}_j)_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(t)}}$
        Identify the largest element of $|\boldsymbol{\Delta}|$: $j_t = argmax_j |\boldsymbol{\Delta}_j|$
        Update $\boldsymbol{\beta}^{(t)}$ in the direction of $j_t$: $\boldsymbol{\beta}_{j_t}^{(t+1)} = \boldsymbol{\beta}_{j_t}^{(t)} + \epsilon\ sign(\boldsymbol{\Delta}_{j_t})$
```

Algorithm 1 produces a sequence of coefficient estimates $\mathbf{B} = \{\boldsymbol{\beta}^{(0)}, \ldots, \boldsymbol{\beta}^{(T)}\}$ which define a path through the p-dimensional parameter space for the coefficients. Variable selection is achieved by "early stopping", i.e., by selecting an element of $\mathbf{B}$ for which some of the coefficients remain at zero (i.e., were never updated by the iterative boosting procedure). This step can employ holdout data, cross-validation, direct model scoring (via, e.g., the AIC or BIC), depending on the problem in question. The primary purpose of boosting techniques (and penalization methods) is to identify a set of candidate models from among a very large number of potential models; the hope is that at least some of these candidate models will have small mean squared prediction error (MSPE). We will emphasize this point later in arguing that the loss function used to calculate the MSPE need not play a central role in identifying a “good” set of candidate models.

2.2.1 EEBoost

Most existing variable selection procedures, whether based on penalization or boosting, focus on regression models which apply to relatively “clean” data, i.e., where outcomes are independent, completely observed, not subject to measurement error, etc. However, there is a vast and ever-expanding toolbox of regression techniques which accommodate these various types of “dirty” data. Many of these techniques avoid specifying a likelihood as the data characteristics being accommodated (e.g., correlation) may be poorly understood and not amenable to modeling. For such techniques, estimation typically involves solving a set of estimating equations.

As an alternative, Wolfson (2011) introduced EEBoost, an extension of the boosting algorithm applicable to problems where coefficient estimation is carried out by solving an estimating equation. The key to EEBoost is that estimating equations, while not exactly corresponding to the gradient of a loss function, often behave much like gradients and hence can take their place in a boosting algorithm. The predictors are scaled to have mean 0 and variance 1. In the rare instance of identical gradients, one of the variables with the tied max gradient could be selected at random to be updated. In the following iteration, it is then very unlikely that the gradient for that variable would again be tied with the others. Algorithm 2 presents EEBoost; note that the vector of estimating equations $\mathbf{g}(\mathbf{X}, \boldsymbol{\beta})$ takes the place of the gradient $|\partial L(\boldsymbol{X}, \boldsymbol{\beta})|/\partial\boldsymbol{\beta}$ from Algorithm 1.

By making use of estimating equations which account for important features of the data, EEBoost aims to produce paths containing coefficient estimates which yield smaller MSPE. Since there is no explicit loss function to minimize, the technique used

Algorithm 2 EEBoost

```
procedure EEBOOST
Set $\boldsymbol{\beta}^{(0)}$ to the zero $p$-vector $\mathbf{0}_p$.
    for $t = 0, \ldots, T$ do
        Compute the estimating equations at the current estimate $\boldsymbol{\beta}^{(t)}$: $\boldsymbol{\Delta} = \mathbf{g}(\mathbf{X}, \boldsymbol{\beta})_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(t)}}$
        Identify the largest element of $|\boldsymbol{\Delta}|$: $j_t = argmax_j|\boldsymbol{\Delta}_j|$
        Update $\boldsymbol{\beta}^{(t)}$ in the direction of $j_t$: $\boldsymbol{\beta}_{j_t}^{(t+1)} = \boldsymbol{\beta}_{j_t}^{(t)} + \epsilon\ sign(\boldsymbol{\Delta}_{j_t})$
```

to generate the variable selection path may not be directly linked to the procedure employed to select the point on that path which minimizes MSPE. For example, it can be shown that when observations are correlated within clusters, accounting for the correlation in estimation of regression parameters yields a smaller MSPE, even though the form of the MSPE does not acknowledge the correlated nature of the data. Hence, in this setting, applying EEBoost with the Generalized Estimating Equations produces variable selection paths which contain coefficient estimates yielding smaller MSPE than a standard LASSO approach which ignores correlation.

As an added benefit, EEBoost is also much faster than competing penalized estimating equation-based techniques, as it does not require solving constrained optimization problems. Wolfson (2011) reported computational speedups of up to 100-fold over existing methods.

2.2.2 Diversifying variable selection paths

The primary goal of EEBoost is to identify a set of candidate models (i.e., a sequence of regression coefficient estimates), $\mathbf{B}$, whose predictive performance can be assessed using external data, cross-validation, or other model scoring techniques. The hope is that there exists at least one $\boldsymbol{\beta}^{(k)} \in B$, say $\beta^{(k^*)}$, such that $|R(\boldsymbol{\beta}^{(k^*)}) - R(\boldsymbol{\beta}_0)| \leq \delta$ for some acceptably small δ. In other words, the path $\mathbf{B}$ must pass "close enough" to the true $\boldsymbol{\beta}_0$; no amount of cross-validation or model scoring can find a suitable $\boldsymbol{\beta}$ in $\mathbf{B}$

otherwise.

In certain settings, there are theoretical guarantees that $\mathbf{B}$ will contain a suitable $\boldsymbol{\beta}^{(k^*)}$. For instance, oracle results for several variants of the LASSO (Zou, 2006; Bunea et al., 2007; Van De Geer, 2008; Huang et al., 2013) guarantee that, if the penalty parameter λ_n is suitably chosen as n increases, then the LASSO solution $\hat{\boldsymbol{\beta}}(\lambda_n)$ converges to $\boldsymbol{\beta}_0$. Previous work by Efron et al. (2004); Rosset et al. (2004); Rosset and Zhu (2007) demonstrated the equivalence (as $T \to \infty$ and $\epsilon \to 0$ with $T \cdot \epsilon \to 0$) between boosting and L_1 penalized paths, suggesting that similar results also hold for boosting. For a broad class of estimating equations, EEBoost can be viewed as gradient descent on a projected likelihood (see Wolfson (2011), using results from Small and Wang (2003), for details), and hence EEBoost closely approximates the variable selection path obtained by applying the LASSO to the aforementioned projected likelihood.

Unfortunately, these theoretical results provide limited insight into the real-world performance of boosting methods. Beyond the fact that asymptotic results may not apply with finite samples, in practice one must choose fixed values of the step length, ϵ, and the number of iterations, T. Further, in settings where the loss function is more complex (e.g., projected likelihoods), existing oracle inequalities may not be applicable. In such cases, it is not clear that the boosting algorithms will yield good variable selection paths. We therefore propose a generalization of the EEBoost algorithm which allows it to generate a wide variety of variable selection paths by setting values of a single threshold parameter.

2.2.3 ThrEEBoost: Thresholded EEBoost

Algorithms 1 and 2 update one coefficient at each iteration, corresponding to the largest element of the gradient or estimating equation. Hence, if $j_t = \arg\max_j \boldsymbol{\Delta}_j$ is unique at each step, $\boldsymbol{\beta}^{(K)}$ can have at most K nonzero entries. Friedman (2004) proposed a generalization of boosting called Thresholded Gradient Descent Regularization (TGDR) wherein multiple elements of the coefficient vector $\boldsymbol{\beta}^{(K)}$ can be updated at each iteration. The elements to be updated correspond to the largest gradient values; how large the gradient needs to be for the corresponding coefficient to be updated is determined by a threshold parameter $\tau \in [0, 1]$. Specifically, given scaled predictors, coefficients are updated if $|\boldsymbol{\Delta}_j| \geq \tau \cdot \max_j |\boldsymbol{\Delta}_j|$. $\tau = 0$ corresponds to updating every coefficient at every iteration, while $\tau = 1$ is equivalent to the original boosting algorithm, assuming that the entries of $\boldsymbol{\Delta}$ are distinct.

We apply this idea to EEBoost, yielding ThrEEBoost, presented in Algorithm 3. Each value of τ yields a distinct coefficient path, $\mathbf{B}(\tau)$. Further, for a fixed value of τ, the computational burden of ThrEEBoost is no higher than EEBoost. When using cross-validation to select the optimal value of τ, ThrEEBoost will be a factor of K times more computationally expensive, where K is the number of thresholding values that are chosen.

Algorithm 3 ThrEEBoost

```
procedure ThrEEBoost
Set $\boldsymbol{\beta}^{(0)} = \mathbf{0}$
    for $t = 0, \ldots, T$ do
        Compute $\boldsymbol{\Delta} = \mathbf{g}(\mathbf{X}, \boldsymbol{\beta})_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(t-1)}}$
        Identify $J_t = \{j : |\boldsymbol{\Delta}_j| \geq \tau \cdot \max_j |\boldsymbol{\Delta}_j|\}$
        for all $j_t \in J_t$ do
            Update $\boldsymbol{\beta}^{(t)}_{j_t} = \boldsymbol{\beta}^{(t-1)}_{j_t} + \epsilon \ \mathrm{sign}(\Delta_{j_t})$
```

2.2.4 Selecting the best model

In standard applications of boosting and EEBoost, the algorithm is run for a predetermined number of iterations, producing a variable selection path from which one chooses the model (i.e., set of coefficient estimates) yielding the smallest MSPE, $\frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} [y_{ij} - x_{ij}\beta^{(t)}]^2$. The process is analogous to solving a LASSO problem for a sequence of values of the penalty parameter λ, then choosing the optimal value of λ.

The ThrEEBoost procedure involves repeating this process for different settings of the threshold parameter τ, yielding a family of variable selection paths indexed by τ. While applying ThrEEBoost with multiple τ values increases the number of coefficient sets for which MSPE must be estimated, it poses no conceptual challenges. In practice, we recommend the following algorithm to choose τ via cross-validation, minimizing the MSPE.

Algorithm 4 Model Selection for ThrEEBoost

```
procedure Cross Validation
    Divide the observations into $K$ folds where $\frac{1}{K}$ of the observations are used as a test set.
    for $k = 1, \ldots, K$ do
        Apply ThrEEBoost for several values of $\tau$.
        Obtain the minimum MSPE of each candidate model on the test set.
        Select the $\tau_k$ that minimizes MSPE.
    Repeat across the $K$ possible test sets and compute the mean of the selected $\tau_k$'s.
```

If cross-validation is computationally infeasible, then a model scoring criterion such as the QIC (Pan, 2001) can also be used: Assuming $Q()$ is the quasi-likelihood, R is the working correlation structure, D is the data (X, Y), Ω_I is the observed information, and $\hat{V}_r$ is the sandwich variance estimate:

$$QIC(R) = -2Q(\hat{\boldsymbol{\beta}}(R); I, D) + 2 * trace(\hat{\Omega}_I \hat{V}_r)$$

Both approaches are illustrated as part of the simulation study in Section 2.3. Cross validation is preferred and is utilized in the data application in Section 2.4.

2.3 Simulation Study

Simulations were conducted in R version 3.2.0 (R Core Team, 2015) using the `threeboost` package provided in the supplementary materials. The code for conducting this simulation study is also available in the supplementary materials.

2.3.1 Sparse regression model with correlated outcomes

We simulated data for $n = 30$ individuals with four correlated observations from each individual. A vector of covariates X_{ij} of length 50 was generated for each individual from a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma}_X$ where $\mathrm{Var}(X_{ijk}) = 0.25$ and for each $\mathrm{Corr}(X_{ijk},\, X_{ijl}) = 0.0$, 0.3, 0.5, and 0.7 $\forall\ k \neq l$. Each correlation level yielded similar results for all of our performance metrics, so we will focus our results on the scenario where $\mathrm{Corr}(X_{ijk},\, X_{ijl}) = 0.3$. The outcome variables for each individual Y_{ij}, $i = 1, \ldots, 30$, $j = 1, \ldots, 4$, were generated from a multivariate normal distribution with mean $\boldsymbol{\mu}_i = \boldsymbol{X}_i\boldsymbol{\beta}$, with an exchangeable correlation matrix such that $\mathrm{Var}(Y_{ij}) = 1$, $\mathrm{Corr}(Y_{ij}, Y_{ik}) = \rho$, $\forall\ j \neq k$. The true values of the coefficient vector $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_{50})$ were set as:

$$\beta_m = \begin{cases} 0.5, 1 \leq m \leq 2 \\ 0.2, 3 \leq m \leq 5 \\ 0.0, 6 \leq m \leq 50 \end{cases}$$

Models which accommodate correlated data are generalized linear mixed models (GLMMs) and marginal models estimated via generalized estimating equations (GEE). GLMMs may be sensitive to assumptions about the distribution of the outcome and random effects. Variable selection techniques for GLMMs typically require maximizing the penalized likelihood and selecting both random and fixed effects (Schelldorfer et al., 2014), which can be computationally demanding. GEE provides an approach to estimation which is more robust to misspecification of the variance; however, existing approaches for variable selection with GEE (Johnson et al., 2008) are based on solving a set of penalized estimating equations, which is also computationally expensive. For this simulation, ThrEEBoost was performed using GEE,

$$\mathbf{g}(\boldsymbol{\beta}) = \sum_{\mathbf{i=1}}^{\mathbf{30}} \boldsymbol{X}_{\mathbf{i}}^{\prime} \boldsymbol{V}_{\mathbf{i}}^{-\mathbf{1}} (\mathbf{Y}_{\mathbf{i}} - \boldsymbol{X}_{\mathbf{i}}^{\prime} \boldsymbol{\beta})$$

where $\boldsymbol{V}_i = \boldsymbol{A}_i^{1/2} \boldsymbol{R}_i(\rho) \boldsymbol{A}_i^{1/2}$ with $\boldsymbol{A}_i = \text{diag}(\text{Var}(\mathbf{Y}_i))$ and $\boldsymbol{R}_i(\rho)$ is the working correlation matrix. For these simulations, we assumed an exchangeable working correlation matrix such that $\boldsymbol{R}_i = (1-\rho)\boldsymbol{I} + \rho\mathbf{1}\mathbf{1}^{\prime}$. ρ was estimated at each iteration via a method of moments estimator using the current value $\boldsymbol{\beta}^{(t)}$ at iteration t.

For each combination of $\rho = \{0.0, 0.3, 0.6\}$ and $\tau = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0, \tau_{CV}\}$, we generated 1000 datasets as outlined above and ran 500 ThrEEboost iterations, producing a variable selection path $\{\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^k\}$ for $k = 1, \ldots, 500$ for each simulated dataset. We selected τ_{CV} using cross-validation using $K = 10$ folds. We estimated MSPE at each point on a path by estimating the average MSE across 100 datasets generated under the same assumptions used to generate the original data.

Table 2.1 shows the minimum MSPE, minimum QIC, number of iterations to reach the minimum MSPE, and variable selection sensitivity and specificity across the 1000 simulations for each combination of ρ and τ. Sensitivity and specificity are given by

$$\text{Sensitivity} = \frac{\sum_{m=1}^{p} |\text{sign}(\hat{\beta}_m^k)|}{\sum_{m=1}^{p} |\text{sign}(\beta_m^{\text{true}})|}, \quad \text{Specificity} = \frac{\sum_{m=1}^{p} 1 - |\text{sign}(\hat{\beta}_m^k)|}{\sum_{m=1}^{p} 1 - |\text{sign}(\beta_m^{\text{true}})|}$$

where $\text{sign}(\beta) = 0$ if $\beta = 0$.

For some simulation runs, the ThrEEBoost algorithm led to a sequence of coefficient estimates which began to alternate between 2 models before finding a solution that uniquely minimized the MSE. These are easy to detect and can be remedied in practice

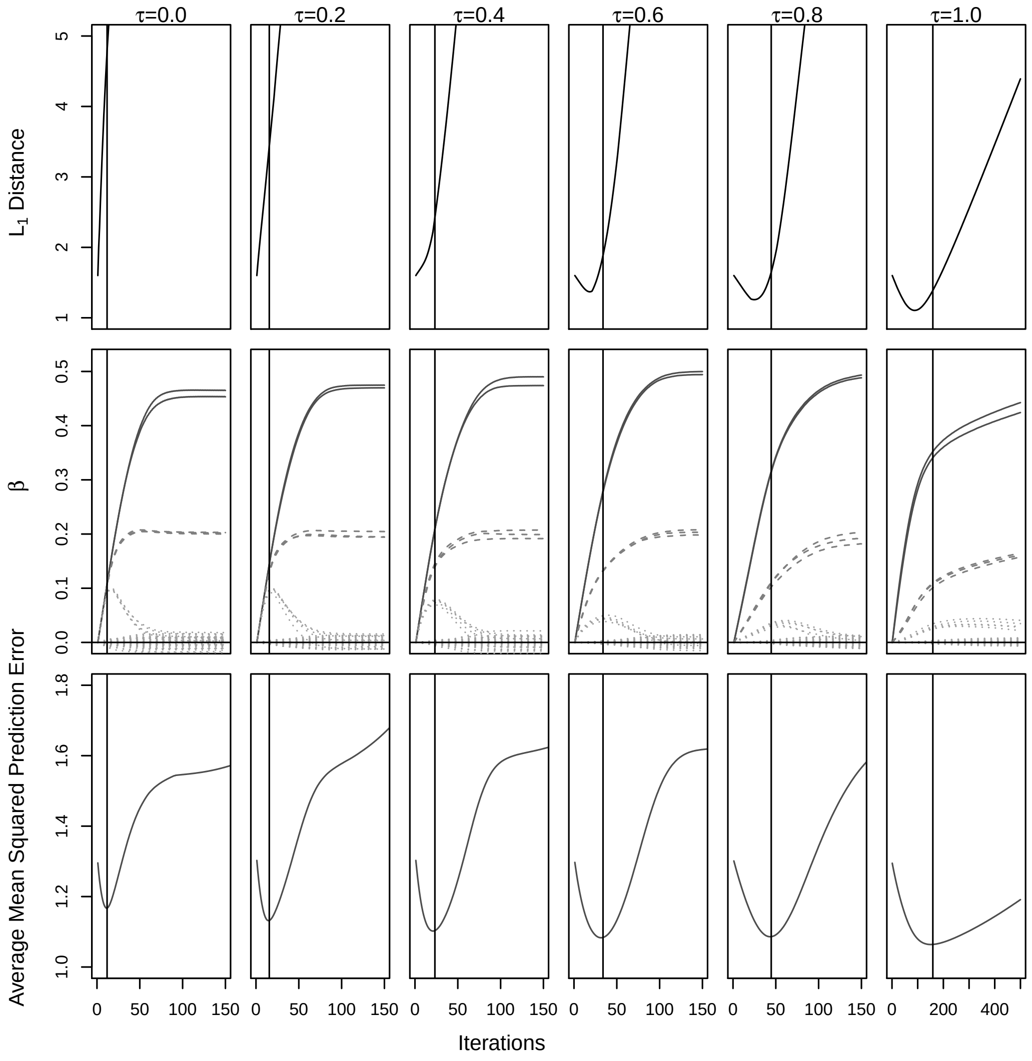

Figure 2.1: Average L_1 distances from the true β (top row), estimated coefficient values (middle row) and MSPE (bottow row) across iterations for various values of τ, when data are generated from a very sparse true regression model with an intra-individual correlation of $\rho = 0.3$. The solid, dashed, and dotted lines in the coefficient plots (middle row) represent coefficients with true values of 0.5, 0.2, and 0.0 respectively. Results are based on 1000 simulations, each with 500 ThrEEBoost iterations. The solid vertical lines show the iteration where the minimum mean squared error is achieved in each scenario.

ρ	$\tau = 0.0$	$\tau = 0.2$	$\tau = 0.4$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 1.0$	$\tau = \tau_{CV}$
	(a) Mean Minimum Prediction Error						
0.0	1.17	1.13	1.09	1.08	1.07	1.07	1.09
0.3	1.16	1.12	1.09	1.06	1.06	1.06	1.08
0.6	1.15	1.11	1.07	1.06	1.05	1.06	1.06
	(b) Median Sensitivity						
0.0	1.00	1.00	1.00	0.80	0.80	0.80	0.80
0.3	1.00	1.00	1.00	0.80	0.80	0.80	1.00
0.6	1.00	1.00	1.00	1.00	0.80	0.80	1.00
	(c) Median Specificity						
0.0	0.00	0.36	0.64	0.80	0.84	0.87	0.76
0.3	0.00	0.31	0.62	0.78	0.82	0.87	0.76
0.6	0.00	0.29	0.60	0.76	0.82	0.87	0.73
	(d) Mean Iterations to Minimum Prediction Error (IQR)						
0.0	11 (9, 13)	15 (12, 17)	22 (17, 26)	34 (25, 41)	44 (34, 53)	158 (125, 183)	32 (21, 41)
0.3	12 (10, 14)	16 (13, 17)	23 (18, 27)	34 (26, 40)	45 (36, 52)	159 (129, 180)	32 (22, 42)
0.6	14 (10, 15)	18 (14, 21)	26 (20, 32)	36 (28, 43)	45 (38, 54)	160 (132, 184)	35 (26, 44)
	(e) Minimum Mean QIC						
0.0	212	199	181	169	162	155	175
0.3	210	199	179	165	160	153	173
0.6	210	197	177	169	160	156	175
	(f) Proportion of simulations with numerical instability						
0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.6	0.01	0.01	0.00	0.01	0.01	0.01	0.01

Table 2.1: Mean minimum prediction error (a), median variable selection (b) sensitivity and (c) specificity, (d) mean number of iterations (25th and 75th percentile) to attain minimum prediction error, (e) minimum mean QIC, and (f) proportion of simulations where algorithm did not find a unique minimum MSE for ThrEEBoost in the sparse true model under different values of the threshold, τ and correlation between intra-individual observations, ρ. Results are based on 1000 simulations, each with 500 iterations.

by selecting another thresholding value. The proportion of simulation runs resulting in numerical instability are reported in part (f) of Tables 2.1 and 2.2.

For each value of ρ, mean minimum MSPE decreased as τ increased from 0.0 to 0.6.

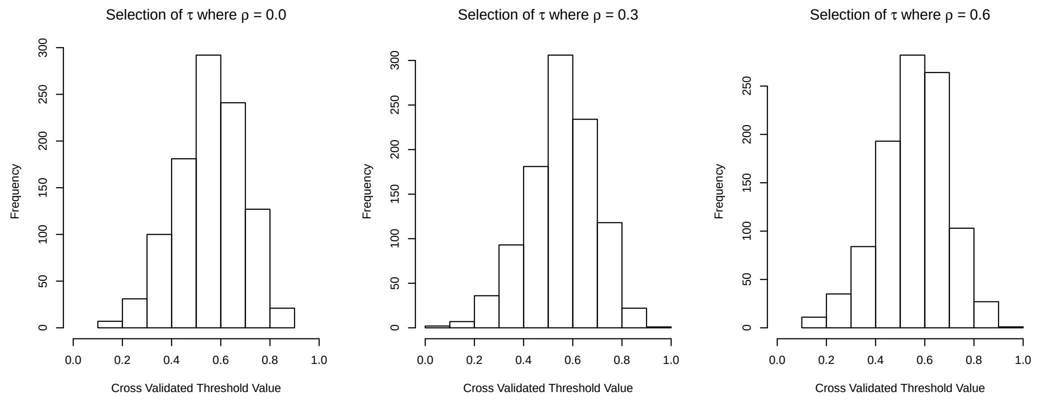

Figure 2.2: The distribution of selected τ values via cross-validation. For each value of ρ, the median τ_{CV} selected was 0.58.

However, values of $\tau \geq 0.6$ resulted in very similar MSEs. ThrEEBoost had similar median sensitivity to EEBoost across values of ρ. For each value of ρ, the sensitivity ranged from 0.80 to 1.00. Specificity increased with τ, ranging from 0 for $\tau = 0$ to 0.87 for $\tau = 1$. ThrEEBoost with $\tau < 1$ reached minimum MSPE with considerably fewer iterations than with $\tau = 1$ (i.e., EEBoost). On average, ThrEEBoost with $\tau < 1$ located the point on the variable selection path achieving minimum MSPE in 3.5 to 14.4 times fewer iterations than EEBoost. Minimum mean QIC decreased as τ increased. Figure 2.1 shows the average L_1 distance from the true β, coefficient values, and MSE across the iterations of ThrEEBoost for different τ values in the scenario where ρ=0.3. Using cross-validation to select an optimal thresholding value τ_{CV}, all three cases chose a median τ of 0.58. The distribution of the chosen τ values are shown in figure 2.2. The results followed the same patterns for each simulated value of ρ and for each of Corr(X_{ijk}, X_{ijl})=0.0, 0.3, 0.5, and 0.7.

2.3.2 Less sparse regression model with correlated outcomes

Next, we undertook an additional simulation study using the same setup as described in the previous section but with a less sparse true regression model for the mean defined by:

$$\beta_m = \begin{cases} 0.5, 1 \leq m \leq 15 \\ 0.2, 16 \leq m \leq 25 \\ 0.0, 26 \leq m \leq 50 \end{cases}$$

Note that the number of nonzero regression coefficients (25) was nearly equal to the number of independent individuals (30). Due to the reduced sparsity of the model, we increased the number of iterations to 1500 for each of 1000 simulated datasets.

Table 2.2 summarizes the MSPE, QIC, sensitivity, specificity, number of iterations to find minimum MSPE, and rate of numerical instability of the algorithm. For all three settings of the correlation parameteter ρ, mean minimum MSPE and QIC both showed a clear "U"-shaped pattern across τ. MSPE achieved the lowest value at $\tau = 0.4$, with $\tau = 0$ and $\tau = 1$ yielding MSPE values 6-28% higher than this minimum value. The optimal τ value to minimize QIC varied from 0.4 to 0.8 depending on ρ. The sensitivity and specificity results show the trade-off that is at play: sensitivity decreases and specificity increases as τ goes from 0 to 1. In this case, specificity improves dramatically up to $\tau = 0.4$ but does not improve substantially with larger τ values; and sensitivity declines steadily but modestly until $\tau = 0.6$. Figure 2.3 shows the L_1 distance from the true $\boldsymbol{\beta}$, the coefficient traceplots, and MSPE across iterations. Figure 2.5 shows the mean QIC across τ values of 0, 1, and τ_{CV} for the various ρ values. The results followed the same pattern for $\rho = 0$ and $\rho = 0.6$. Results were also similar in scenarios where the pairwise correlation between covariates was set to 0, 0.5, and 0.7 (data not shown).

ρ	$\tau = 0.0$	$\tau = 0.2$	$\tau = 0.4$	$\tau = 0.6$	$\tau = 0.8$	$\tau = 1.0$	$\tau = \tau_{CV}$
	(a) Mean Minimum Prediction Error						
0.0	1.95	1.78	1.65	1.77	2.02	2.12	1.65
0.3	1.53	1.45	1.36	1.42	1.53	1.63	1.35
0.6	1.82	1.71	1.73	1.78	1.86	1.88	1.74
	(b) Median Sensitivity						
0.0	1.00	0.96	0.92	0.88	0.84	0.80	0.92
0.3	1.00	0.96	0.92	0.88	0.88	0.88	0.92
0.6	1.00	1.00	0.92	0.92	0.92	0.88	0.96
	(c) Median Specificity						
0.0	0.00	0.24	0.56	0.64	0.68	0.72	0.52
0.3	0.00	0.24	0.56	0.60	0.64	0.64	0.52
0.6	0.00	0.24	0.56	0.60	0.60	0.68	0.52
	(d) Mean Iterations to Minimum Prediction Error (IQR)						
0.0	40 (40, 51)	43 (44, 50)	49 (49, 58)	63 (59, 79)	88 (73, 116)	696 (213, 966)	53 (49, 59)
0.3	47 (46, 52)	46 (46, 51)	52 (51, 58)	68 (63, 79)	102 (93, 120)	845 (871, 980)	55 (50, 61)
0.6	43 (45, 54)	42 (46, 51)	45 (49, 58)	59 (58, 76)	89 (83, 117)	767 (834, 1000)	53 (49, 59)
	(e) Minimum Mean QIC						
0.0	340	314	295	317	354	378	296
0.3	334	319	287	287	299	391	282
0.6	470	470	460	458	418	536	487
	(f) Proportion of simulations with numerical instability						
0.0	0.14	0.10	0.04	0.07	0.10	0.12	0.07
0.3	0.02	0.02	0.01	0.01	0.01	0.03	0.01
0.6	0.03	0.02	0.02	0.02	0.04	0.03	0.03

Table 2.2: Mean minimum prediction error (a), median variable selection (b) sensitivity and (c) specificity, (d) mean number of iterations (25th and 75th percentile) to attain minimum prediction error, (e) minimum mean QIC, and (f) proportion of simulations where algorithm did not find a unique minimum MSE for ThrEEBoost in the less sparse true model under different values of the threshold, τ, and correlation between intra-individual observations, ρ. Results are based on 1000 simulations, each with 1500 ThrEEBoost iterations.

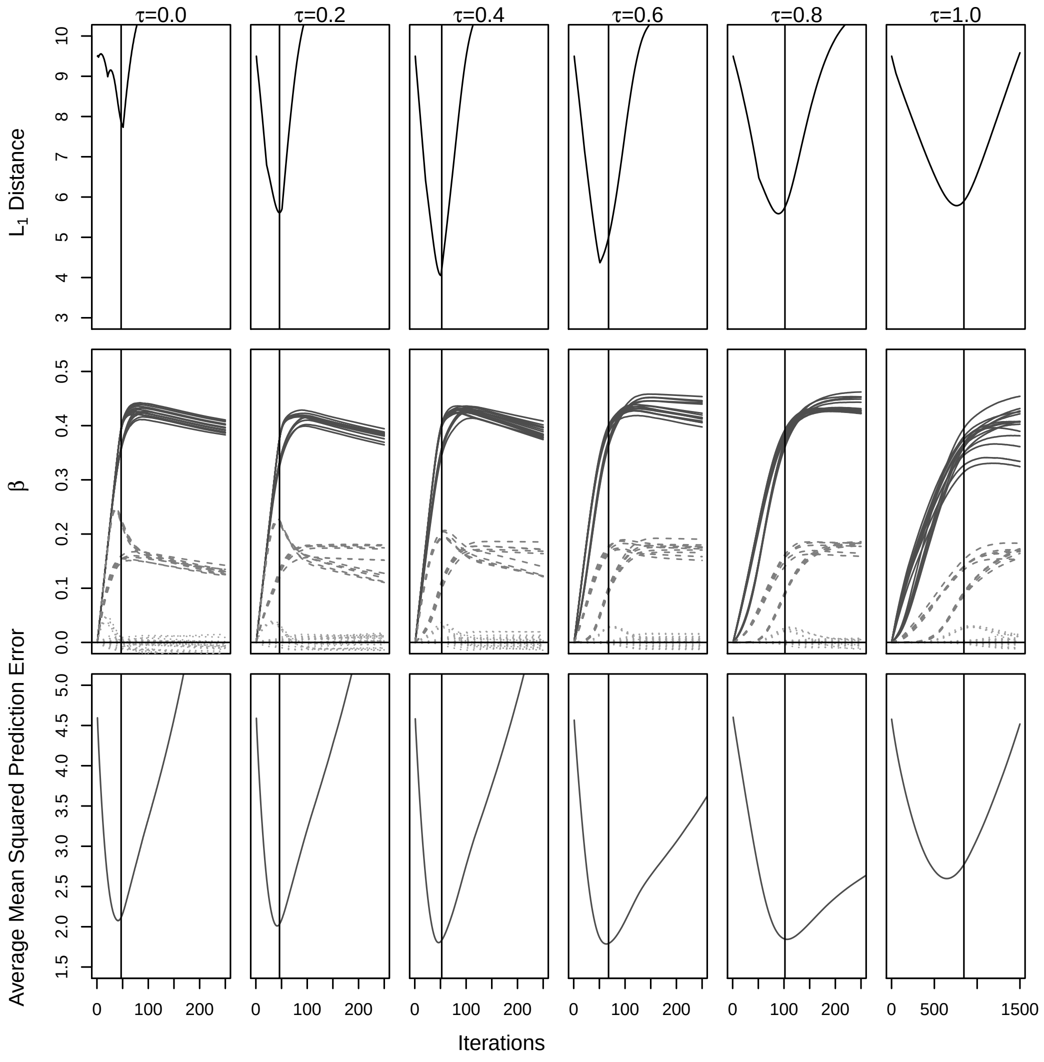

Figure 2.3: Average L_1 distances from the true β (top row), estimated coefficient values (middle row) and MSPE (bottow row) across iterations for various values of τ, when data are generated from a less sparse true regression model with an intra-individual correlation of $\rho = 0.3$. The solid, dashed, and dotted lines in the coefficient plots (middle row) represent coefficients with true values of 0.5, 0.2, and 0.0 respectively. Results are based on 1000 simulations, each with 1500 ThrEEBoost iterations. The solid vertical lines show the iteration where the minimum mean squared error is achieved in each scenario.

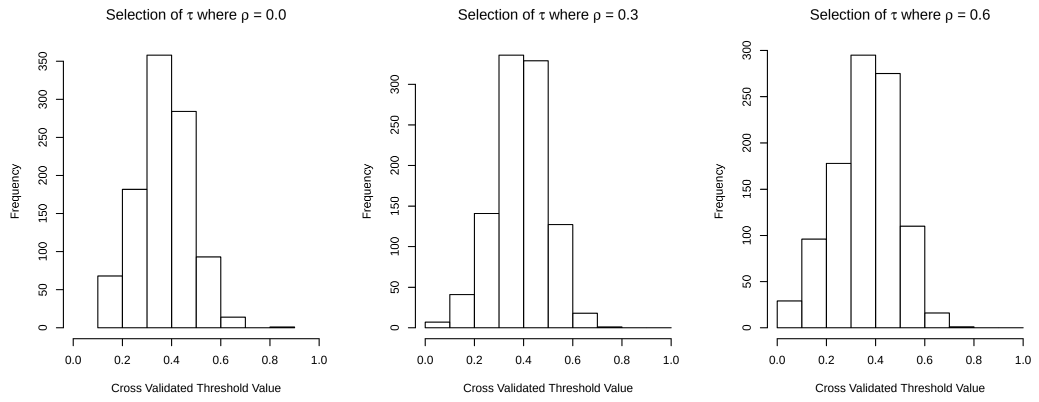

Figure 2.4: The distribution of selected τ values via cross-validation. For each value of ρ, the median τ_{CV} selected were 0.38, 0.40, and 0.38.

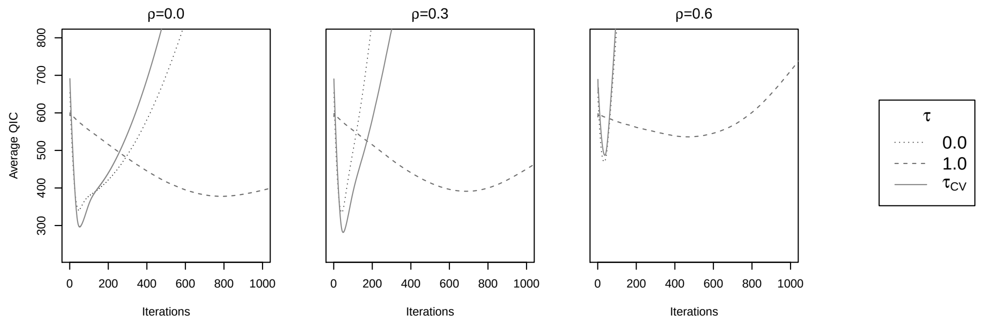

Figure 2.5: Average QIC when data are generated from a less sparse true regression model with an intra-individual correlation of $\rho = 0.3$. Results are based on 1000 simulations, each with 1500 ThrEEBoost iterations.

Using cross-validation to select τ offered an improvement over EEBoost (i.e., ThrEEBoost with $\tau = 1$). The MSPE shrunk by about 22%, 18%, and 7% for the cases where ρ=0.0, 0.3, and 0.6, respectively. The median τ selected was lower than in the sparse case with values of 0.38, 0.40, and 0.38, respectively. The distributions of τ_{CV} are shown in Figure 2.4.

2.4 Data application - Box Lunch Study

We illustrate the application of ThrEEBoost to outcome data from the Box Lunch Study, a randomized controlled trial to evaluate the effect of portion size availability on caloric intake and weight gain (French et al., 2014). Two hundred and thirty-three eligible individuals were randomized to one of four groups: three "free lunch" groups and a "no free lunch" group which served as a control. The three "free lunch" conditions differed according to the number of calories provided in the daily box lunch: 400, 800, and 1600.

	Coefficients	
Variable	ThrEEBoost	LASSO
Race (Black)	0.27	0.24
Race (Hispanic)	0.24	0.35
Health (1=exc 5=poor)	0.17	0.08
Age	0.17	0.11
Lost control past 28 days	0.15	–
Education (HS)	0.14	0.14
Have fridge at work	0.12	0.19
TFEQ Disinhibition	0.10	0.32
Lbs gain before you noticed	0.06	0.16
Dissatisfied with weight	–	0.20
Light actvty min/day (251-2100)	–	0.15
Freq fast food (0=never 5=7+ times/week)	–	0.06
Limit food you eat	–	0.05
Marital status (Married)	-0.088	-0.11
Moderate activity min/day (2101-5900)	–	-0.05
Frequency self-weigh (0=never 5=every day)	–	-0.08
Freq restaurant/week	–	-0.10
TFEQ Hunger	–	-0.16

Table 2.3: Coefficients for the optimal ThrEEBoost ($\tau = 0.4$) and LASSO models selected by cross-validated MSE. Small coefficients (magnitude < 0.05) are omitted. "–" indicates that the variable was not selected in the model.

Here, we explore the factors associated with BMI in the "no free lunch" group consisting of $n = 49$ individuals on whom BMI measurements were taken at four time

	ThrEEBoost τ						
	0	0.2	0.4	0.6	0.8	1.0	LASSO
CV MSE	0.72	0.78	0.60	0.66	0.66	0.75	0.83

Table 2.4: Estimated mean squared prediction error for ThrEEBoost and LASSO models. (CV MSE) denotes models selected by minimizing cross-validated MSE.

points (baseline, 1, 3, and 6 months). There were 54 covariates of interest, including demographic (e.g. age, gender, race, height, education), lifestyle (e.g. smoking status, physical activity levels), and psychosocial (e.g. frequency of self-weighing, degree of satisfaction with current weight) covariates recorded at baseline, and a variety of longitudinally-recorded food-related outcomes such as average daily caloric intake and average daily servings of fruits and vegetables. The outcome and predictors were scaled to have zero mean and unit variance prior to analysis.

ThrEEBoost was applied using the Gaussian Generalized Estimating Equations with an exchangeable working correlation structure. The algorithm was run for $\tau = 0, 0.2, 0.4, 0.6, 0.8$, and 1, and the optimal model for each τ was selected as the one which minimized the MSPE estimated by five fold cross-validation. The smallest MSPE overall (0.60) was achieved by ThrEEBoost with $\tau = 0.4$. To implement the LASSO, least angle regression (LARS) was utilized over five fold cross-validation to select an optimal penalty parameter which minimized the MSPE. Fitting the optimal LASSO model on the full data set, we obtained MSPE of 0.83. The non-zero coefficients for this model are summarized in Table 2.3, and compared to the coefficients from the LASSO fit with smallest cross-validated MSPE. The models selected by LASSO and ThrEEBoost share some covariates in common, but remain quite distinct. Overall, the ThrEEBoost model is more parsimonious than the LASSO model. Notably, the LASSO estimates relatively large coefficients for some variables (e.g., Dissatisfied with weight) which are not selected by ThrEEBoost. This may be due to the fact that the LASSO ignores the correlated nature of the outcome, and is therefore overly optimistic about the amount of statistical signal present in the data. Figure 2.6 summarizes the coefficients of the optimal ThrEEBoost model for various values of the threshold parameter τ. The estimated coefficients for $\tau = 0.4, 0.6, 0.8$, and 1 are generally similar, with higher τ values leading to slightly more parsimonious models. However, as shown in Table 2.4, these subtle differences can yield very different prediction errors, hence the path diversity offered by ThrEEBoost is an asset.

2.5 Discussion

We have introduced a thresholded extension of the EEBoost algorithm, ThrEEBoost, and critically assessed its operating characteristics in variable selection and prediction in high-dimensional models. We have shown via a detailed simulation study that ThrEEBoost provides a predictive advantage over EEBoost. In cases when the true regression model was relatively sparse, ThrEEBoost required considerably fewer iterations than EEBoost to locate models with comparable performance. When the regression model was less sparse, varying the thresholding parameter in ThrEEBoost allowed for the exploration of a larger set of variable selection paths, leading to the discovery of models with lower MSPE.

Several limitations of the present study should be acknowledged. This simulation study focused solely on cases of normally distributed correlated outcome data, using GEE with an exchangeable working correlation. Further research is needed to clarify the benefits of thresholded variable selection with other correlation structures, and for other classes of estimating equations. Second, while the numerical experiments are promising, we have not provided theoretical results that guarantee, e.g., that ThrEEBoost possesses an oracle property. In ongoing work, we are exploring these theoretical properties of ThrEEBoost and clarifying its relationship to "hybrid" penalized variable selection procedures such as the elastic net.

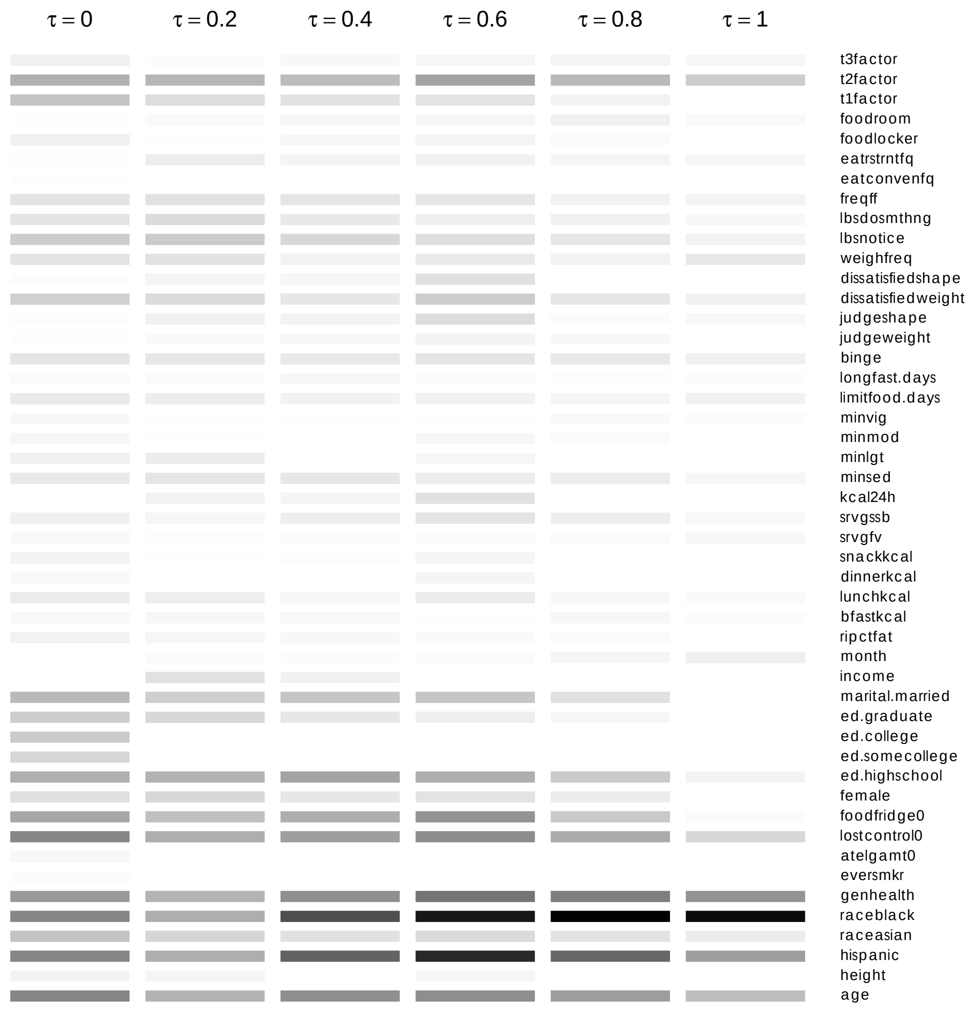

Figure 2.6: Coefficient magnitudes for the optimal models (chosen by cross-validated MSE) for different values of τ. Each row corresponds to a different variable; darker shades of gray correspond to higher coefficient magnitudes. The names of the variables are displayed on the right; a data dictionary giving the variable descriptions is provided in the Supplementary Materials.

2.6 Supplementary Materials

Simulation R Code: The R code for the simulation study. (sim_threeboost.R, R file)

Data Application R code: The R code to analyze data from the Box Lunch Study. (BLS analysis 06-09-16.R, R file)

ThrEEBoost R Package: The R package which implements ThrEEBoost is available on the Comprehensive R Archive Network (CRAN). The most up-to-date version of the `threeboost` package is available at `https://www.github.com/jwolfson/threeboost`. (threeboost-master.zip, zip archive)

Chapter 3

MEBoost

3.1 Introduction

Variable selection is a well-studied problem in situations where covariates are measured without error. However, it is common for covariate measurements to be error-prone or subject to random variation around some mean value. Consider, for instance, a study wherein subjects report their daily food intake on the basis of a dietary recall questionnaire. There is variation from day to day in an individual's calorie consumption, but it is also well established in the nutrition literature that there is error associated with the recall or measurement of the number of calories in a meal (Spiegelman et al., 1997; Fraser and Stram, 2012). In the usual regression setting, ignoring measurement error leads to biased coefficient estimation (Rosner et al., 1992), and hence the presence of measurement error has the potential to affect the performance of variable selection procedures.

There has been relatively little research done about variable selection in the presence of measurement error. Sørensen et al. (2012) introduced a variation of the Lasso that allows for Normal, i.i.d., additive covariate measurement error. Datta and Zou (2017) proposed the convex conditioned Lasso (CoCoLasso) which corrects for both additive and multiplicative measurement error in the normal case. Both of these methods are applicable to linear models for continuous outcomes, but do not easily extend to regression models for other outcome types (e.g., binary or count data). Meanwhile, there is a sizable statistical literature on methods for performing estimation and inference for low-dimensional regression parameters in the presence of measurement error (Rosner et al., 1992; Stefanski and Carroll, 1985; Fuller, 1987), but these approaches do not address the variable selection problem and cannot be applied in large p, small n problems.

We propose a novel method for variable selection in the presence of measurement error, MEBoost, which leverages estimating equations that have been proposed for low-dimensional estimation and inference in this setting. MEBoost is a computationally efficient path-following algorithm that moves iteratively in directions defined by these estimating equations, only requiring the calculation (not the solution) of an estimating equation at each step. As a result, it is much faster than alternative approaches involving, e.g., a matrix projection calculation at each step. MEBoost is also flexible: the version that we describe is based on estimating equations proposed by Nakamura (Nakamura, 1990), which apply to some generalized linear models, and the underlying MEBoost algorithm can easily incorporate measurement error-corrected estimating equations for other regression models. We conducted a simulation study to compare MEBoost to the Convex Conditioned Lasso (CoCoLasso) proposed by Datta and Zou (2017) and the "naive" Lasso which ignores measurement error. We also applied MEBoost to data from the Box Lunch Study, a clinical trial in nutrition where caloric intake across a number of food categories was based on self-report and hence measured with error.

3.2 Background

3.2.1 Regression in the Presence of Covariate Measurement Error

Our discussion of measurement error models draws heavily from Fuller (1987). When modeling error the covariates can be treated as random or fixed values. Structural models consider the covariates to be random quantities and functional models consider the covariates to be fixed (Buonaccorsi, 2010). We consider a structural model. Let $\mathbf{Y} = \mathbf{X}\beta + \epsilon$, where $\mathbf{X}$ is a (random) matrix of covariates of dimension $n \times p$, β a vector of coefficients of length p, ϵ is a vector of Normally distributed i.i.d. random errors of length n, and $\mathbf{Y}$ is the resultant outcome vector also of length n. In an additive measurement error model, we assume that what is observed is not $\mathbf{X}$ but rather the "contaminated" or "error-prone" matrix $\mathbf{W} = \mathbf{X} + \mathbf{U}$ where $\mathbf{U}$ a random $n \times p$ matrix.

When a model is fit that ignores measurement error, i.e. it assumes that the true model is $\mathbf{Y} = \mathbf{W}\beta_{\mathbf{W}} + \epsilon$, the resulting estimates $\hat{\beta}_{\mathbf{W}}$ are said to be *naive* and satisfy

$$E[\hat{\beta}'_{\mathbf{W}}] = \beta'(\mathbf{\Sigma}_{\mathbf{XX}} + \Delta)^{-1}\mathbf{\Sigma}_{\mathbf{XX}} \tag{3.1}$$

where β is the true coefficient vector, $\mathbf{\Sigma}_{\mathbf{XX}}$ is the covariance matrix of the covariates and $\Delta \equiv \Sigma_{UU}$ is the covariance matrix of the measurement error. In the case of linear regression with a single covariate, (3.1) simplifies to an attenuating factor that biases the coefficient estimates towards zero. However, with multiple covariates the bias may increase, decrease, and even change the sign of the estimated coefficients. Notably, measurement error affecting a single covariate can bias coefficient estimates in all of the covariates, even those that are not measured with error (Buonaccorsi, 2010).

3.2.2 Variable selection in the Presence of Measurement Error

Ma and Li (2010) presented methods to account for measurement error while performing variable selection in parametric and semi-parametric settings. Focusing on the parametric setting, they proposed a wide scoping method that can be used in more than just generalized linear models. The method relies on deriving the full likelihood of each observation and it's corresponding score function, $S^*_{eff}(\mathbf{W}_i, \mathbf{Y}_i, \beta)$, choosing a penalty function and finding its derivative, $p'(\beta)$, then solving the penalized estimating equations:

$$\sum_{i=1}^{n} S^*_{eff}(\mathbf{W}_i, \mathbf{Y}_i, \beta) - np'(\beta) = 0 \tag{3.2}$$

Solving the penalized equations can be very difficult computationally, especially in the high dimensional setting. Therefore, we will look to compare our method with faster methods that are variants of the Lasso, which can be solved much more quickly.

3.2.3 Lasso in the Presence of Measurement Error

Sørensen et al. (2012) analyze the Lasso (Tibshirani, 1996) in the presence of measurement error by studying the properties of

$$\hat{\beta}_{Lasso,\lambda_n} = \underset{\alpha}{\operatorname{argmin}} \left(||\mathbf{Y} - \mathbf{W}\alpha||_2^2 + \lambda_n ||\alpha||_1 \right). \tag{3.3}$$

$\hat{\beta}_{Lasso,\lambda_n}$ is asymptotically biased when $\lambda_n/n \to 0$ *as* $n \to \infty$ since $E[\hat{\beta}'_{Lasso,\lambda_n}] = \beta'(\mathbf{\Sigma_{XX}} + \Delta)^{-1}\mathbf{\Sigma_{XX}}$. Notice this is the same bias that is introduced when naive linear regression is performed on observed covariates. Sørensen et al. (2012) derive a lower bound on the magnitude of the non-zero coefficient elements below which the corresponding covariate will not be selected, and an upper bound on the L_1 estimation error $||\hat{\beta}_{\mathbf{W}} - \beta||_1$. They show that with increasing measurement error the lower bound increases, i.e., increasing measurement error adds non-informative noise to the system and so for the signal associated with the relevant covariates to be identified the signal must increase. Increased measurement error also leads to an increase in the upper bound of the estimation error. Sign consistent selection is also impacted by the presence of covariate measurement error. Sørensen et al. (2012) set a lower bound on the probability of sign consistent selection in this setting. The result requires that the *Irrepresentability Condition with Measurement Error* (IC-ME) holds. The IC-ME requires that the measurements of the relevant and irrelevant covariates have limited correlation, relative to the size of the relevant measured covariate correlation. Note the sample correlation of the irrelevant covariates is not considered. By studying the form of the lower bound, it can be concluded that (at least when using the Lasso) measurement error introduces a greater distortion on the selection of irrelevant covariates than it does in the selection of relevant covariates.

Sørensen et al. (2012) introduced an iterative method to obtain the Regularized Corrected Lasso with constraint on the radius R:

$$\hat{\beta}_{RCL} = \underset{||\beta||_1 < R}{\arg\min}\{||y - W\beta||_2 - n\beta'\Delta\beta + \lambda||\beta||_1\}. \tag{3.4}$$

The main results of their simulation study were consistent with their analytical results, namely that the corrected Lasso had a slightly lower selection rate for the true covariates than the naive Lasso, but was also more conservative in including irrelevant covariates. Further, the prediction error, as measured by both $||\hat{\beta} - \beta||_1$ and $||\hat{\beta} - \beta||_2$, was lower for the corrected Lasso.

The major drawback of the corrected Lasso method is that it is very computationally intensive, involving an iterative calculation where each step involves a projection of an updated $\hat{\beta}$ onto the L_1-ball for a given radius R. The iterative process must be conducted for each fixed value of the radius R. The selected values of R provide a path of possible solutions for $\hat{\beta}_{RCL}$. Hence, the approach seems impractical for large-scale problems and for repeated application in a simulation study.

3.2.4 The Convex Conditioned Lasso (CoCoLasso)

A recent paper by Datta and Zou (2017) proposes an alternative approach which they refer to as the Convex Conditioned Lasso (CoCoLasso). Consider the following reformulation of the Lasso problem,

$$\hat{\beta_L}(\lambda) = \arg\min_{\beta} \frac{1}{2}\beta^{'}\Sigma\beta - \rho^{'}\beta + \lambda||\beta||_1. \tag{3.5}$$

The CoCoLasso is based on the Loh and Wainwright (2012) corrections for the predictor-outcome correlation ρ and variance matrix Σ in the presence of measurement error. When error-prone covariates $\mathbf{W}$ are measured in place of $\mathbf{X}$, we can get corrected estimates $\tilde{\rho}$ and $\hat{\Sigma}$:

$$\tilde{\rho} = \frac{1}{n}\mathbf{W}^{'}\mathbf{Y} \qquad\qquad \hat{\Sigma} = \mathbf{W}^{'}\mathbf{W} - \Delta \tag{3.6}$$

where Δ is the (assumed known) variance in the measured $\mathbf{W}$. These estimators are unbiased. A measurement error corrected Lasso estimate could then be derived by substituting $\tilde{\rho}$ and $\hat{\Sigma}$ into (3.5). The problem with this idea is that the corrected matrix $\hat{\Sigma}$ may not be a valid covariance matrix, since it is possible to be non positive semi-definite. If $\hat{\Sigma}$ has a negative eigenvalue, then this Lasso function would be non-convex and unbounded. To overcome this obstacle, the key to the CoCoLasso (Datta and Zou, 2017) is calculating the projection of $\hat{\Sigma}$ onto the space of positive definite matrices:

$$(\hat{\Sigma})_+ = \arg\min_{\Sigma \geq 0} ||\hat{\Sigma} - \Sigma||_{max}. \tag{3.7}$$

The CoCoLasso then solves a standard Lasso problem in which $\hat{\Sigma}$ and ρ with the corrected values from (3.6) and (3.7), yielding the CoCoLasso estimator:

$$\hat{\beta_C}(\lambda) = \arg\min_{\beta} \frac{1}{2}\beta^{'}(\hat{\Sigma})_+\beta - \tilde{\rho}^{'}\beta + \lambda||\beta||_1. \tag{3.8}$$

When $\hat{\Sigma}$ is not positive definite, the projection from (3.7) can be challenging to compute. However, the projection only needs to be done once, unlike the Sørensen et al. (2012) correction which requires a projection at each iteration.

3.3 MEBoost: Measurement Error Boosting

Our proposed variable selection algorithm, MEBoost (**M**easurement **E**rror **Boost**ing), is based on an iterative functional gradient descent type algorithm that generates variable selection paths. The key idea is that, instead of following a path defined by the gradient of a loss function (e.g., the likelihood), the "descent" follows the direction defined by an estimating equation $\mathbf{g}(\mathbf{Y}, \mathbf{X}, \beta)$. The algorithmic structure of MEBoost is shared with ThrEEBoost (Thresholded Estimating Equation Boost, Brown et al. (2017)), a general-purpose technique for variable selection based on estimating equations. While ThrEEBoost described an approach to performing variable selection in the presence of correlated outcomes by leveraging the Generalized Estimating Equations (Liang and Zeger, 1986), MEBoost achieves improved variable selection performance in the presence of measurement error by following a path defined by a measurement error corrected score function due to Nakamura which is described in Section 3.3.1. Nakamura's approach is applicable to linear regression models with normal additive or multiplicative measurement error. Closed-form corrected score functions are also derived for Poisson, Gamma, and Wald regression. Nakamura comments that no closed form correction can be created for logistic regression. By using this family of corrected score functions, the MEBoost algorithm is more broadly applicable than the corrected Lasso and CoCoLasso, neither of which is obviously generalizable beyond linear regression.

3.3.1 Corrected Score Function

Nakamura (1990) proposed a set of corrected score functions for performing estimation and inference in the generalized linear regression model where covariates are subject to additive measurement error with known variance matrix Δ. In general, the corrected score function S* based on the covariates measured with error (**W**), has the expectation equal to the score function, S, based on the true covariates (**X**). For the normal linear model, Nakamura proposed the following correction to the negative log-likelihood to account for measurement error:

$$l^*(\mathbf{Y},\mathbf{W},\beta)^{'} = -\frac{n}{2}log(2\pi) - nlog(\sigma) - \frac{1}{2\sigma^2}\sum[(y_i - \beta' w_i)^2 - \beta'\Delta\beta] \tag{3.9}$$

Differentiating (3.9) with respect to β, we obtain the corrected score function:

$$S^*(\mathbf{Y},\mathbf{W},\beta)^{'} = S(\mathbf{Y},\mathbf{W},\beta)^{'} + n\sigma^{-2}\beta^{'}\Delta. \tag{3.10}$$

In this case the corrected score function is the 'naive' score function, $S(\mathbf{Y},\mathbf{W},\beta)^{'}$, with a measurement error correction determined by the sample size, model error, measurement errors, and the coefficient value: $n\sigma^{-2}\beta^{'}\Delta$. The naive score function is the score function from the true model calculated with the measured covariates:

$$S(\mathbf{Y},\mathbf{W},\beta,\sigma) = \sigma^{-2}\left(\mathbf{W}^{'}\mathbf{Y} - \mathbf{W}^{'}\mathbf{W}\beta\right) \tag{3.11}$$

The corrected variance estimate will be calculated as $\partial l^*/\partial\sigma = 0$, which in the normal case is:

$$\hat{\sigma}^2 = n^{-1}\left(\mathbf{Y} - \mathbf{W}\beta^*\right)^{'}\left(\mathbf{Y} - \mathbf{W}\beta^*\right) - \beta^{*'}\Delta\beta^*. \tag{3.12}$$

Similarly to the corrected score function, the corrected variance estimate is the naive variance estimate, $n^{-1}\left(\mathbf{Y}-\mathbf{W}\beta^*\right)'\left(\mathbf{Y}-\mathbf{W}\beta^*\right)$, with a measurement error correction. The correction reduces the estimated variance, thus subtracting the noise introduced by the measurement error. In the variance case the correction factor is determined only by the true coefficient vector and the measurement error variance.

As another example, the correction for Poisson distributed data is the following:

$$S(\mathbf{Y},\mathbf{W},\beta)=\sum[y_k w_k-(w_k-\Delta\beta)exp(\beta' w_k-\beta'\Delta\beta/2)] \tag{3.13}$$

which we apply in our data application (see Section 3.5). Nakamura (1990) also provides corrections for multiplicative measurement error in linear regression, as well as measurement error in Gamma and Wald regression. In what follows, we use the normal linear additive measurement error corrected score function as part of an iterative path-following algorithm that performs variable selection in the presence of covariate measurement error.

3.3.2 The MEBoost Algorithm

Our proposed variable selection algorithm, MEBoost, consists of applying ThrEEBoost with the corrected score function and corrected variance estimate described in the previous section. Algorithm 5 summarizes the MEBoost procedure.

Let $\tau \in [0,1]$ be the fixed thresholding parameter. Starting with a β estimate of $\mathbf{0}$ and a $\hat{\sigma}^2=1$, the corrected score function $\mathbf{S}^*$ is calculated at these values, and the magnitude of each component of $\nu \equiv \mathbf{S}^*$ is recorded. The indices of elements to update are identified by a thresholding rule, $J_t=\{j : |\nu_j| \geq \tau \cdot \max_j|\nu_j|\}$. The next point in the variable selection path, $\beta^{(1)}$, is obtained by adding a small value, γ, to each of these elements in the direction corresponding to the signs of each ν_j for $j \in J_t$. This updated $\beta^{(1)}$ is used to calculate an updated corrected $\sigma^{2(1)}$. The algorithm continues for T iterations, where T is typically chosen to be large (e.g., 1,000).

Algorithm 5 MEBoost

```
procedure MEBoost
Set $\boldsymbol{\beta}^{(0)} = \mathbf{0}$
Set $\sigma^{2,(t=0)} = 1$
    for $t = 0, \ldots, T$ do
        Compute $\nu = \mathbf{S}^*(\mathbf{Y}, \mathbf{W}, \boldsymbol{\beta})_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(t-1)}}$
        Identify $J_t = \{j : |\nu_j| \geq \tau \cdot \max_j |\nu_j|\}$
        for all $j_t \in J_t$ do
            Update $\boldsymbol{\beta}^{(t)}_{j_t} = \boldsymbol{\beta}^{(t-1)}_{j_t} + \gamma \ \mathrm{sign}(\nu_{j_t})$
        Set $\sigma^{2,(t)} = n^{-1}\left(\mathbf{Y} - \mathbf{W}\beta^{(t)}\right)'\left(\mathbf{Y} - \mathbf{W}\beta^{(t)}\right) - \beta^{(t)\prime}\Delta\beta^{(t)}$
```

The parameters γ, T and τ interact to determine the specific variable selection path that results from the algorithm. The smaller the value of γ the smaller the distance between β estimates on the selection path, while a larger value of γ leads to larger jumps in the selection path. Ideally, a very small value of γ (e.g., 0.01). would be used, but if $||\beta||_{\mathbf{1}}$ is large, a large number of iterations, T, may be required to generate a selection path. This of course is the trade-off one is required to make when determining the step size. A selection path incremented by only a small value is preferable to a path which takes large steps, but the time required for a large number of iterations may become prohibitive. With each of the t iterations those elements of the coefficient vector that are still of size zero have not been selected at this iteration. A conservative selection approach takes a combination of small γ and T, whereas a more aggressive approach takes a combination of larger value γ and T. In the case when $\tau = 1$, the MEBoost algorithm only updates the element(s) with the maximum absolute value. For any combination of γ and T, this is the most conservative approach that can be taken and will lead to sparser models than when a threshold is considered. It also requires a much larger value of T.

The parameter τ determines how many coefficients are updated at each iteration; it offers a compromise between updating each coefficient at every iteration ($\tau = 0$, similar to standard gradient descent) and updating only the coefficient corresponding to the element of the estimating equation with largest magnitude ($\tau = 1$). In the context of Generalized Linear Models without measurement error, Wolfson (2011) showed that setting $\tau = 1$ yields an update rule that is asymptotically equivalent (as $T \to \infty$, $\gamma \to 0$, and $T \cdot \gamma \to 0$) to following the path of minimizers of an L_1-penalized projected artificial log-likelihood ratio whose tangent is the GLM score function. In the case when $\tau = 1$, the MEBoost algorithm only updates the element(s) with the maximum absolute value. For any combination of γ and T, this is the most conservative approach that can be taken and will lead to sparser models than when a threshold is considered. It also requires a much larger value of T. By allowing multiple directions to be updated at each iteration, MEBoost can explore a much wider range of variable selection paths; as we discuss later, cross-validation can be used to select the parameter τ which leads to the optimal level of thresholding. In ThrEEBoost (Brown et al., 2017), it was shown that a threshold in the range of 0.4-0.8 may perform better than thresholds closer to 0 or 1.

Connection to the ME-Lasso

Nakamura's measurement corrected score functions are derivatives of corrected negative log likelihoods. In the normal case, the correction is exactly that described in Sorensen et al. (see Equation (3.3)). Hence, the arguments of Rosset et al. (2004) can be applied to show that 1) MEBoost applied with S^* and threshold value $\tau = 1$, and 2) the solutions to (3.3), have the same local behavior. Specifically, under some regularity conditions (see Wolfson (2011)), as $T \to \infty$ and $\epsilon \to 0$ with $T \cdot \epsilon \to 0$, MEBoost's iterative steps match the sequence of solutions to (3.3).

Selecting a final model

For a fixed τ, identifying a final model involves choosing a point on the variable selection path generated by Algorithm 5; this is akin to choosing the penalty parameter in the Lasso. Cross-validation using a loss function relevant to the problem at hand (e.g., mean squared error) can be used to select a $\hat{\beta}$ on the path. Cross-validation can similarly be used to select the best value of τ. The full procedure is described in Algorithm 6.

Algorithm 6 Model Selection for MEBoost

```
procedure CROSS VALIDATION
    Divide the observations into $K$ folds where $\frac{1}{K}$ of the observations are used as a test set.
    for $k = 1, \ldots, K$ do
        Apply MEBoost for fixed value $\tau$.
        Obtain the mean squared prediction error of each candidate model on the test set.
        Calculate $||\hat{\beta}||_1^{(k)}$ for the $\hat{\beta}^{(t)}$ that minimizes mean squared prediction error.
    Repeat across the $K$ possible test sets and compute the mean of the selected $||\hat{\beta}||_1^{(k)}$'s.
```

3.4 Simulation Study

To examine the impact of measurement error in the covariates on variable selection we performed a simulation study. We evaluated MEBoost by comparing it to two variable selection methods: the Convex Conditioned Lasso (CoCoLasso), and the “naive” Lasso which does not correct for measurement error.

3.4.1 Simulation Set-up

Data were generated from a linear regression model with iid normal errors, $\mathbf{Y} = \mathbf{X}\beta + \epsilon$; where $\epsilon_\mathbf{i} \sim N(\mathbf{0}, \sigma_\epsilon^2)$ and $\sigma_\epsilon = 1.5$. The sample size for all studies is 80. The true covariates are drawn from a multivariate normal distribution, $\mathbf{X} \sim MVN(\mathbf{0}, \Sigma_{XX})$.

Σ_{XX} is a block diagonal matrix with diagonal entries equal to 1, and 10 by 10 blocks corresponding to a group of 10 covariates with an exchangeable correlation structure with common pairwise correlation $\phi = 0.3$. In all simulations the true model has 10 non-zero coefficients and 90 zero coefficients, i.e., $\beta = (\mathbf{1}_{10}, \mathbf{0}_{90})$, so that the relevant covariates in the first block were correlated.

The measured covariates were generated as $\mathbf{W} = \mathbf{X} + \mathbf{U}$ for $\mathbf{U}$ a matrix whose columns were generated as described below. To explore the impact of different types of measurement error, we considered 10 different scenarios for generating the columns of U and varying the assumptions made about it. In the first five scenarios, $\mathbf{U}$ is assumed to be normally distributed with mean zero and covariance matrix Ω, and the scenarios explore different structures for Ω. In each of Scenarios 1-5, we correctly specify the distribution of $\mathbf{U}$ when applying MEBoost and the CoCoLasso. Scenarios 6-10 explore cases where the distribution of $\mathbf{U}$ is incorrectly specified.

1. Base case: $\mathbf{U} \sim N(\mathbf{0}, \delta^2\Omega_1)$, where $\Omega_1 = \mathbf{I}$ the identity matrix, and $\delta^2 = 0.75$.

2. Varying δ^2: $\delta_j^2 = 0.3375 + 0.075j$ for j in 1-10. This pattern repeats across the blocks of 10 covariates. The relevant covariates have similar spreads of measurement error to the irrelevant covariates.

3. Correlated Measurement Error: Measurement error was assumed normally distributed with an exchangeable correlation structure Within each block, $\mathbf{U} \sim N(\mathbf{0}, \delta^2\Omega_3)$ where $\Omega_3 = \rho\mathbf{1}\mathbf{1}' + (1-\rho)\mathbf{I}$, $\delta^2 = 0.75$, $\rho = 0.3$, and $\mathbf{1}$ is vector of ones.

4. Varying δ^2 with correlation: $\mathbf{U}$ is distributed normally and is centered at 0 with $\delta_j^2 = 0.3375 + 0.075j$ for j in 1-10. This pattern repeats across the blocks of 10 covariates. The relevant covariates have similar spreads of measurement error to the irrelevant covariates. In blocks of ten, there is correlation in $\mathbf{U}$ of 0.3

5. Some $\mathbf{U}$'s not measured with error: $\mathbf{U} \sim N(0, \delta^2\Omega_5)$, where $\delta^2 = 0.75$, $diag(\Omega_5) = [0, 1, 0, 1, \dots]$ and $\Omega_{5,ij} = 0$ for $i \neq j$.

6. Overestimated δ^2: $\mathbf{U}$ generated as in Scenario 1, but we specify $\delta^2 = 1.5$.

7. Underestimated δ^2: $\mathbf{U}$ generated as in Scenario 1, but we specify $\delta^2 = 0.375$.

8. Misspecified correlation: $\mathbf{U}$ generated as in Scenario 3, but we ignore the correlation and specify $\Omega = \delta^2\mathbf{I}$ in running MEBoost and CoCoLasso.

9. Measurement error is distributed uniformly: Each entry $\mathbf{U}_{ij}$ of $\mathbf{U}$ is generated independently from a Uniform distribution, $\mathbf{U}_{ij} \sim U(-1.5, 1.5)$. MEBoost and CoCoLasso are run assuming $\mathbf{U} \sim N(\mathbf{0}, \delta^2\mathbf{I})$ with $\delta^2 = 0.75 = Var(\mathbf{U}_{ij})$.

10. Measurement error is distributed asymmetrically: Each entry $\mathbf{U}_{ij}$ of $\mathbf{U}$ is generated independently from a shifted exponential distribution, $\mathbf{U}_{ij} + \sqrt{0.75} \sim exp(\sqrt{0.75})$. MEBoost and CoCoLasso are run assuming $\mathbf{U} \sim N(\mathbf{0}, \delta^2\mathbf{I})$ with $\delta^2 = 0.75 = Var(\mathbf{U}_{ij})$.

MEBoost was performed for each threshold value in the set $\tau = (0.0, 0.2, 0.4, 0.6, 0.8, 1.0)$, and cross-validation (using the error-prone covariates) was used to select the optimal value of τ and number of MEBoost iterations, as well as the value of λ in the CoCoLasso and naive Lasso. We compared MEBoost, CoCoLasso, and naive Lasso on two metrics of prediction error: mean squared error based on the true covariates (MSE $= \frac{1}{n}(\mathbf{Y} - \mathbf{X}\hat{\beta})'(\mathbf{Y} - \mathbf{X}\hat{\beta})$), mean squared error prediction based on the measured covariates (MSE-M $= \frac{1}{n}(\mathbf{Y} - \mathbf{W}\hat{\beta})'(\mathbf{Y} - \mathbf{W}\hat{\beta})$). These metrics were estimated using independent test sets generated during each individual simulation. We also computed L_1 distance from the true β, and variable selection sensitivity and specificity. For each scenario the metrics presented are the average over 1,000 simulations, and are calculated at intervals of 0.05 along $||\hat{\beta}||_1 \in \{0.05, 0.1, 0.15, ..., 15\}$; the true value, $||\beta||_1 = 10$. Because the MEBoost algorithm may change multiple indices at each iteration it may not have values along each interval in the path. To account for this, a linear approximation of the relevant statistic was made at each point in the path.

We note that in this simulation study we chose to investigate model performance based on both the true and error-prone covariates. The motivation for techniques like ours which account for measurement error is to uncover the underlying relationship between the error-free covariates $\mathbf{X}$ and the outcome $\mathbf{Y}$. Hence, in an ideal world, values of $\mathbf{X}$ would be available on some subset (or an independent set) of observations so that prediction error could be assessed and the "best" model chosen. However, in practice we will often only have access to the error-prone covariates $\mathbf{W}$ for model fitting. So, if error-free measurements $\mathbf{X}$ are not (and may never be) available, is it worthwhile to correct for measurement error? Buonaccorsi (1995) argued against correction, using the logic that the future predictions will be based on (error-prone) $\mathbf{W}$, not on (error-free) $\mathbf{X}$. Indeed, it can be shown in simple linear regression, that without the correction in a large sample the expected value of MSE-M is less than or equal to that of an estimate ignoring measurement error. However, as seen in the results section that follows, we found that correcting for measurement error decreased prediction error regardless of whether predictions were computed using error-free or error-prone covariates. Since we often only have mismeasured data available, it is reassuring to see that we are able to use the measured covariates to perform cross-validation to select a model that will provide us with an accurate relationship between the outcome and true covariate. This finding is discussed in greater detail below.

3.4.2 Simulation Results

Table 3.1 presents the minimum MSE, MSE-M, L_1 distance from the true β, sensitivity, and specificity at the minimum MSE for the three variable selection methods across the 10 scenarios. In all ten scenarios, MEBoost had the lowest MSE, MSE-M, and L_1 distance from the true β. The CoCoLasso has 16.6%-71.7% higher prediction error from the true covariates than MEBoost and in the case where measurement error is overestimated, the prediction error from the CoCoLasso is 5.26 times that of MEBoost. This is likely due to the fact that the Loh and Wainwright correction $\hat{\Sigma}$ in (3.6) is more negative, and hence requires a "longer" (and hence potentially more distorting) projection onto the space of positive definite matrices.

In terms of variable selection, MEBoost had a greater sensitivity and lower specificity than CoCoLasso in each case while Lasso had the lowest specificity. The Lasso struggles most when correlation is present in the measurement error. The MSE is about 2.5 times that of MEBoost, when we allow MEBoost to account for the correlation. All methods perform poorly when we misspecify Δ by ignoring the correlation. The sensitivity and specificity are at high levels for most simulations with the exception of the misspecified Δ that ignored correlation. Overestimating δ lead to a more conservative selection process with a high specificity, while underestimating δ had a higher sensitivity. The L_1 distance from the true β can also tell us about performance. Again, the scenario where we misspecify Δ by ignoring correlation performs worst.

Figure 3.1 shows how these metrics vary with $||\hat{\beta}||_1$ in Scenario 2 (varying δ); similar plots for the other 9 scenarios appear in the supplementary materials. The MSE, MSE-M, and L_1 distance are U-shaped with a minimum just before $||\hat{\beta}||_1$ reaches the true $||\beta||_1$ of 10. For each metric, MEBoost achieves the lowest values, meaning it is closest to the true vector β and has the best prediction given either true covariates or covariates measured with error. The CoCoLasso performs worst. For sensitivity, MEBoost most quickly goes towards 1. This comes at the cost of specificity, but specificity does not begin to decrease drastically until after $||\hat{\beta}||_1 = 6$. The sensitivity of the Lasso eventually surpasses that of MEBoost, but when it does, the specificity decreases very sharply. The

CoCoLasso at every point has the lowest value of sensitivity and the highest of specificity.

Figure 3.2 summarizes the differences between the variable selection path defined by MEBoost and CoCoLasso, in Scenario 1. We see in the left-hand plot that MEBoost includes nearly all relevant covariates within the first few iterations. The coefficient values are then increased at each step towards their true values of 1; at some point, additional irrelevant covariates begin to enter the model. In contrast, The CoCoLasso tends to increase the magnitude of a single coefficient, $\hat{\beta}_i$. This is confirmed by the middle plot, which shows the proportion of variables which are included in only one of the two models, as $||\hat{\beta}||_1$ increases. We see from this plot that the number of covariates included in one model but not the other decreases as the CoCoLasso identifies more of the relevant covariates, then increases again as each method identifies different sets of irrelevant covariates. The rightmost plot shows the element-wise maximum and mean absolute difference between the estimated coefficient vectors from the two methods. We see that the CoCoLasso is quickly updating fewer elements as the maximum distance between estimates of β_i between the two methods grows initially, while MEBoost updates each of the relevant covariates gradually. As $||\hat{\beta}||_1$ approaches 4, the maximum distance begins to decrease. This is because MEBoost begins updating the index of the largest estimate from the CoCoLasso more quickly than the CoCoLasso is, since it has turned its attention to other variables. Once $||\hat{\beta}||_1$ surpasses the true value of 10, inclusion of irrelevant covariates causes the maximum distance to increase again.

3.5 Data Application

We applied our method to baseline data collected in the Box Lunch Study, a randomized trial of the effects of portion size availability on weight change. In the study, a total of 219 subjects were randomized to one of four groups: in three groups, subjects were provided a free daily lunch with a fixed number of calories (400, 800, and 1600). The control group was not provided a free lunch.

We considered the problem of predicting the number of times subjects reported binging on food in the last month, using Poisson regression with 99 explanatory variables. All variables were measured at baseline. 16 of the 99 explanatory variables were self-reported measures; of these 16, 8 were measures of food consumption and therefore possibly subject to substantial measurement error we will notate δ_D^2. Another 8 may have also been measured with error, notated δ_M^2. Kipnis et al. (2003) examined a nutritional study with a 24 hour recall, and found that the correlation between the true and reported consumption of protein and energy was only 0.336. We assume this relationship exists in each of our variables measured with error. Assuming the measurement error variance $Var(U_i) \equiv \delta_i^2$ is independent of the variance of the true covariate $Var(X_i) \equiv \sigma_{X_i}^2$, we can obtain:

$$\rho_{W_i,X_i} = \rho_{X_i+U_i,X_i} = \frac{\sigma_{X_i}^2}{\sqrt{\sigma_{X_i}^2(\sigma_{X_i}^2 + \delta_i^2)}} \implies Var(W_i) = 1 - \rho_{X_i+U_i,X_i}^2 = \frac{\delta_i^2}{\sigma_{X_i}^2 + \delta_i^2} \tag{3.14}$$

and hence $Var(W_i) = 1 - 0.336^2 = 0.887$. This is the value we will need to provide MEBoost for our assumption of the measurement error. We assume this level of measurement error for each 24 hour dietary recall variable. After scaling our predictors to have zero mean and unit variance, we applied our method with the Nakamura correction. Since our measured data has its variance $(\delta_i^2 + \sigma_{X_i}^2)$ scaled to equal 1, we assumed that the 8 dietary recall covariates measured with error had $\hat{\delta}_D^2 = 0.887$. Since dietary variables may be more prone to measurement error than other variables, we scaled the assumed error of the other 8 variables to be half that of the nutritional variables: $\hat{\delta}_M^2 = \hat{\delta}_D^2/2$. The remaining variables were assumed to be measured without error. We conducted a sensitivity analysis to assess the performance of our method by setting $\hat{\delta}_D^2 = 0.5$ and 0.25.

To select tuning parameters, we employed 8-fold cross validation based on the deviance on a training set consisting of 70% of the data. The performance of our model was evaluated on the remaining test set. We present the models derived from MEBoost performed with three different thresholds τ: 0.2, 0.6 (the approximate value estimated using cross-validation), and 0.9.

Table 3.2 shows the selected variables and estimated prediction error (MSE-M, bottom row) for various MEBoost models along with results from the naive Lasso. We did not compare to the Measurement Error Lasso or the CoCoLasso because implementing these techniques in a problem of this size was computationally infeasible. The deviance and MSE-M were lowest for the model selected by MEBoost assuming the highest measurement error (= 0.887) and a threshold value of 0.6. This model ($\hat{\delta}^2_D = 0.887$ and $\tau = 0.6$) selected just 4 variables, which were a subset of the 7 chosen with the naive Lasso. The other two MEBoost models included up to two additional variables to the MEBoost model that minimized MSE-M (selected with $\hat{\delta}^2_D = 0.887$ and $\tau = 0.6$). Regardless of the assumption about the level of measurement error, using a threshold value of $\tau = 0.2$ leads to the inclusion of several variables with small coefficients, and a much higher deviance and prediction error. Of particular note is that the naive Lasso (and MEBoost with the lower threshold) included the variable corresponding to the number of daily calories consumed at breakfast, while the best-performing MEBoost models (with $\tau = 0.6$ and 0.9) did not. Since it is based on a 24-hour dietary recall, this variable may be particularly susceptible to measurement error induced by recall bias.

3.6 Discussion

We examined the variable selection problem in regression when the number of potential covariates is large compared to the sample size and when these potential covariates are measured with measurement error. We proposed MEBoost, a computationally simple descent-based approach which follows a path determined by measurement error-corrected estimating equations. We compared MEBoost, via simulation and in a real data example, with the recently-proposed Convex Conditioned Lasso (CoCoLasso) as well as the naive Lasso which assumes that covariates are measured without error. In almost all simulation scenarios, MEBoost performed best in terms of prediction error and coefficient bias. The CoCoLasso is more conservative with the highest specificity in each case, but sensitivity and prediction are better with MEBoost. In the comparison of selection paths, we saw that MEBoost was more aggressive in identifying variables to be included in the model more quickly than the CoCoLasso. These differences were most apparent when the measurement error had a larger variance and a more complex correlation structure. Specifically, when faced with a data set of 1000 observations and 1000 covariates, MEBoost obtained a solution in 1.3 seconds, while the CoCoLasso needed 6:17.

As shown in the simulation study, MEBoost has lower prediction error than the Lasso on independent test data when predictions are based on the true (i.e., non-error-prone) covariates. It is interesting to note that MEBoost retains some advantage, albeit a more modest one, over the Lasso when predictions are based on error-prone covariates. This finding appears to contradict the intuition that accounting for covariate measurement error provides no benefit when the goal is prediction and error-free covariates will never be available. However, the observed benefit in our simulation is likely due to the fact that MEBoost is somewhat more flexible than the Lasso as it uses an additional parameter, the threshold τ, which allows it to explore the model space more comprehensively. Nevertheless, it is reassuring that by using the error-prone covariates to perform cross-validation and select a model, MEBoost still allows us to select a model that offers an improvement in prediction in the setting where we will have correctly measured covariates.

MEBoost, while a promising approach, has some limitations. One limitation–which is shared with many methods that correct for measurement error–is that we assume that the covariance matrix of the measurement error process is known, an assumption which in many settings may be unrealistic. In some cases, it may be possible to estimate these structures using external data sources, but absent such data one could perform a sensitivity analysis with different measurement error variances and correlation structures, as we demonstrate in the real data application. Another challenging aspect of model selection with error-prone covariates is that, even if the set of candidate models is generated via a technique which accounts for measurement error, the process of selecting a final model (e.g., via cross-validation) still uses covariates that are measured with error. However, we showed in our simulation study that MEBoost performs well in selecting a model which recovers the relationship between the true (error-free) covariates and the outcome, even when using error-prone covariates to select the final model. This finding suggests that the procedure for generating a "path" of candidate models has a greater influence on prediction error and variable selection accuracy than the procedure picking a final model from among those candidates.

To conclude, we note that while we only considered linear and Poisson regression in this paper, MEBoost can easily be applied to other regression models by, e.g., using the estimating equations presented by Nakamura (1990) or others which correct for measurement error. In contrast, the approaches of Sørensen et al. (2012) and Datta and Zou (2017) exploit the structure of the linear regression model and it is not obvious how they could be extended to the broader family of generalized linear models. The robustness and simplicity of MEBoost, along with its strong performance against other methods in the linear model case suggests that this novel method is a reliable way to deal with variable selection in the presence of measurement error.

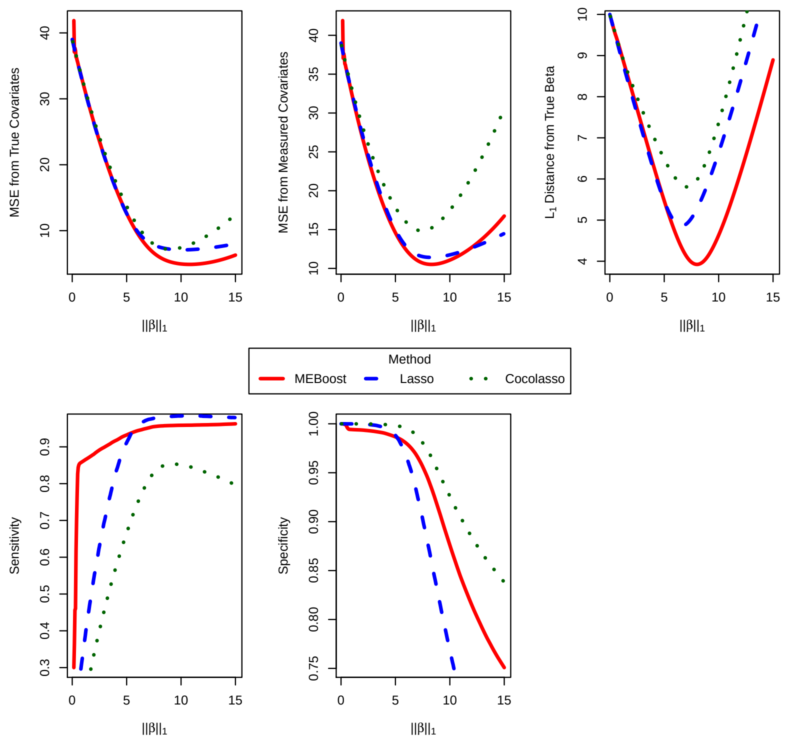

Figure 3.1: Summary statistics for the scenario with varying levels of independent measurement errors. Plots are of MSE, MSE-M, L_1 distance from $\hat{\beta}$ to the true value of β, sensitivity, and specificity across the mean path over 1,000 simulations.

3.7 Tables and Figures

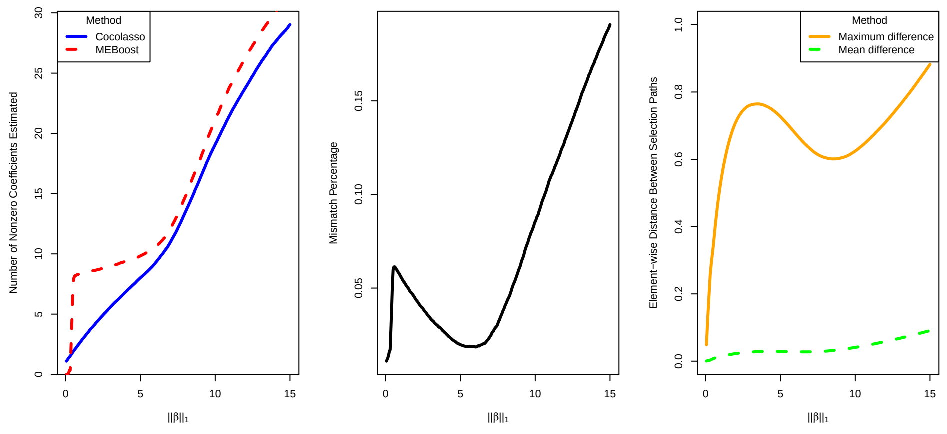

Figure 3.2: In the scenario, with iid measurement error, 1,000 simulations were conducted comparing the variable selection path of MEBoost to CoCoLasso. Plots are of the number of nonzero coefficients included in the model, the portion of coefficients that are 'mismatched' with only one of the two models selecting a certain variable, and the element-wise distance in a coefficient between the two methods.

Scenario	Method	MSE	MSE-M	L_1D	SENS	SPEC
Measurement error iid	MEBoost	4.86	10.65	5.17	0.95	0.86
	Lasso	7.13	11.63	6.75	0.98	0.75
	CoCoLasso	6.30	12.53	6.04	0.92	0.91
Varying δ	MEBoost	4.88	10.52	5.21	0.96	0.85
	Lasso	7.08	11.42	6.76	0.98	0.76
	CoCoLasso	7.18	14.84	6.50	0.85	0.95
Some $\delta = 0$	MEBoost	3.65	6.70	3.57	0.99	0.87
	Lasso	4.88	6.23	5.42	0.99	0.83
	CoCoLasso	6.23	11.15	5.60	0.92	0.95
Varying δ & correlation	MEBoost	6.19	19.12	6.27	0.95	0.87
	Lasso	15.18	21.42	9.63	0.80	0.79
	CoCoLasso	8.94	22.47	7.77	0.78	0.94
Correlation in measurement error	MEBoost	6.16	19.27	6.29	0.95	0.87
	Lasso	15.78	22.24	9.75	0.80	0.79
	CoCoLasso	8.67	22.29	7.81	0.78	0.94
Overestimated δ	MEBoost	4.00	10.21	3.46	0.94	0.94
	Lasso	7.18	11.71	6.75	0.98	0.76
	CoCoLasso	21.05	28.22	9.11	0.37	1.00
Underestimated δ	MEBoost	5.54	10.81	6.28	0.98	0.76
	Lasso	7.18	11.71	6.75	0.98	0.76
	CoCoLasso	6.46	12.02	6.20	0.95	0.87
Misspecified Δ, ignores correlation	MEBoost	12.79	21.03	9.41	0.86	0.80
	Lasso	15.78	22.24	9.75	0.80	0.79
	CoCoLasso	15.67	24.17	9.51	0.55	0.93
Measurement error from asymmetric distribution	MEBoost	4.89	10.68	5.16	0.95	0.85
	Lasso	7.21	11.85	6.81	0.98	0.75
	CoCoLasso	7.32	15.20	6.71	0.85	0.95
Measurement error from uniform distribution	MEBoost	4.81	10.52	5.17	0.96	0.84
	Lasso	7.19	11.75	6.80	0.99	0.75
	CoCoLasso	6.61	13.91	6.26	0.87	0.94

Table 3.1: Performance metrics for the 1,000 simulations in various measurement error scenarios. The models were selected at the point with minimum MSE-M.

	$\hat{\delta}_D^2 = 0.887$			$\hat{\delta}_D^2 = 0.5$			$\hat{\delta}_D^2 = 0.25$			Naive
Variable	$\tau = 0.2$	$\tau = 0.6$	$\tau = 0.9$	$\tau = 0.2$	$\tau = 0.6$	$\tau = 0.9$	$\tau = 0.2$	$\tau = 0.6$	$\tau = 0.9$	Lasso
Ate lg amt past 28 days	0.09	0.23	0.39	0.08	0.23	0.33	0.09	0.22	0.29	0.21
Lost control past 28 days	0.09	0.23	0.21	0.08	0.23	0.22	0.09	0.22	0.23	0.24
TFEQ Disinhibition	0.09	0.23	0.09	0.08	0.23	0.18	0.09	0.22	0.21	0.17
BCT: Max clicks for pizza slice	0.09	0.21	0.1	0.08	0.23	0.19	0.09	0.22	0.2	0.16
Long fast (0=no days 6=every day)	0.09	-	-	0.08	-	-	0.09	-	-	-
Judge your shape (0=not at all 6=markedly)	0.07	-	-	0.07	0.06	-	0.07	0.06	-	-
Judge your weight (0=not at all 6=markedly)	0.07	-	-	0.07	-	0.06	0.07	0.06	0.06	0.06
Dissatisfied with shape (0=not at all 6=markedly)	0.07	-	-	0.07	-	-	0.07	-	-	-
BCT: Rounds of clicking for pizza	0.06	-	-	0.06	-	-	0.07	-	-	-
BCT: Pmaxpizza/(Pmaxpizza+Pmaxread)	0.06	-	-	0.06	-	-	0.06	-	-	-
TFEQ Hunger	0.06	-	-	0.06	-	-	0.06	-	-	-
Dissatisfied with weight (0=not at all 6=markedly)	0.06	-	-	0.06	-	-	0.06	-	-	-
NDSR breakfast kcals at BL	0.06	-	-	0.05	-	-	-	-	-	0.08
CDRS body image (1=thinnest 9=fattest)	0.05	-	-	0.05	-	-	0.05	-	-	-
Dem9 Household income	-	-	-	-	-	-	-	-	-	-0.05
Eat lunch in cafeteria	-	-	-	-	-	-	-0.05	-	-	-
Eat items from home, days/wk	-0.06	-	-	-0.06	-	-	-0.06	-	-	-
Cohort first lunch date	-0.07	-	-	-0.07	-	-	-0.08	-	-	-
Deviance	210.43	89.06	100.49	222.07	90.22	93.23	208.07	90.63	90.38	97.12
MSE-M	22.35	5.61	8.10	23.28	5.69	7.29	22.04	5.97	6.21	7.89

Table 3.2: Coefficients, Deviance, and MSE-M from selected models for MEBoost with specified value of τ and $\hat{\delta}_D^2$ and the Lasso. Small coefficients (magnitude < 0.05) are omitted. "-" indicates that the variable was not selected in the model.

Chapter 4

SuperBoost

4.1 Introduction

Most techniques for variable selection in regression models (e.g., gradient descent (Friedman et al., 2000), Lasso (Tibshirani, 1994), Elastic Net (Zou and Hastie, 2005), L_1-regularization (Park and Hastie, 2007), and for a Cox proportional hazard model (Fan and Li, 2002)) assume that the data satisfy key modeling assumptions. For example, it is usually assumed that outcomes are independent, that covariates are measured without error, and that no data values are missing. Recent research has focused on the problem of extending variable selection to settings where one of these assumptions may be violated. For example, there is the CoCoLasso (Datta and Zou, 2017) and MEBoost (Brown et al., 2017) for covariates with measurement error, Penalized GEE (Johnson et al., 2008) and GEEBoost (Brown et al., 2017) and for correlated outcomes, and Yang et al. (2005) for missing data.

In the low-dimensional setting, some work has been done in estimation in the presence of multiple data challenges. In the instance of censored data and correlation, Pike and Weissfeld (2013) used a joint model approach to analyze longitudinal data on mortality and morbidity. It relies on the conditional longitudinal model, $L(T|u)$, and survival model, $S(Y|u)$, being independent given a random effect, u, distributed according to $F(u)$. This gives us the full likelihood.

$$\mathcal{L}(Y,T,\theta) = \prod_{i=1}^{n} \int_{-\infty}^{\infty} L_i(Y|u) S_i(T|u) dF(u) \tag{4.1}$$

Since this can be written in the form of a likelihood, using penalization to perform variable selection would be straightforward using a coordinate-wise descent algorithm.

Buonaccorsi et al. (2000) proposed a method to account for measurement error in a mixed model for longitudinal data. This is an extension of his work in regression calibration for uncorrelated outcomes. This model includes a random effect and the coefficients can be corrected for additive or multiplicative measurement error. Further, Li et al. (2005), faced a longitudinal data set with a surrogate in place of the true outcome. The surrogate was assumed to be measured with error that was not additive nor multiplicative. Instead, the surrogate was count data, which followed a structure so that the measurement could not be negative; the likelihood was truncated at 0. Therefore, they needed to adapt Buonaccorsi's method to add a latent variable for the measured variable with a point mass at 0.

As far as we are aware, there is no general technique to perform variable selection when standard regression assumptions are violated. We propose two novel variable selection techniques for this situation. We propose SuperBoost, which combines multiple iterative variable selection algorithms which each address a single data characteristic post-hoc. We also propose AltBoost which combines addresses each data challenge alternatively in iterations. We start with a literature review of methods accounting for multiple data assumption violations as well as ways to combine models together. Next we introduce our methods, building upon the ideas of boosting and stacking. We present the algorithm for our method SuperBoost. Then we conduct a simulation to evaluate our methods. Lastly, we discuss the results limitations of these methods.

4.2 Literature Review

4.2.1 Variable selection when regression assumptions are violated

While there has been some work in data with multiple regression assumptions violated in the low dimensional setting, less research has been conducted in the high-dimensional setting where variable selection is required. The adaptive Lasso (He et al., 2015) can estimate random effects in a mixed model for survival data. These are the same challenges we discussed with Pike and Weissfeld (2013), however this is focused on variable selection. This is done in two stages. First, penalization is used to perform variable selection. Second, the selected covariates are used without penalization to reduce the bias of the fixed effect estimates. Koch, B.; Vock, V.; Wolfson (2017) developed Group Lasso and Doubly Robust Estimation (GLiDeR), an expansion of the Group Lasso (Yuan and Lin, 2006). GLiDeR estimates the average causal effect while performing variable selection for clustered data, which is the same thing Cefalu et al. (2016) did with model averaged double robust (MA-DR) estimators. The estimators are robust to a misspecification in either the propensity score or the model for the outcome. It is not meant to correct for both issues simultaneously.

Ni et al. (2010) proposed a method for variable selection in semi-parametric mixed models. They utilize a double penalty on the likelihood to penalize the regression coefficients as well as the roughness of non-parametrics components of the model. This is done in the iterative procedure where we obtain coefficient estimates using ridge regression (Hoerl and Kennard, 1970), then estimate the variance of the random effects and roughness of the nonparametric components using restricted maximum likelihood. The method performs well as selecting the fixed effects, but doesn't incorporate selecting the random effects into the model. It is also limited to normal outcomes and cannot accomodate other likelihood structures.

Each of these methods has its advantages. They perform well when addressing one or two specific regressions assumption violations. However, none have the flexibility to address violations of another type quickly. Particularly when the data violate several key assumptions, it may be challenging to posit a model which simultaneously accounts for all these violations. In such cases, it may be useful to combine multiple variable selection procedures, each of which accounts for a different characteristic of the data generating process. The hope is that an "omnibus" variable selection method based on combining a number of more "targeted" methods would improve variable selection performance.

This idea of aggregating results from multiple models or methods is not new, and goes under many names in the literature, including model averaging (Hoeting et al., 1999), ensemble learning (MacKay, 1995), stacking (Wolpert, 1992), and super learning (Van Der Laan et al., 2007). A handful of papers have applied model averaging ideas to the variable selection problem (e.g. (Viallefont et al., 2001), (Symonds and Moussalli, 2011)), but they generally focus on combining regression models that rely on the same underlying assumptions. Others (Díaz et al., 2015) have used super learning to compute variable importance metrics on the basis of combining a variety of machine learning models, but these approaches do not actually perform variable selection and cannot be applied when the number of variables is large compared to the sample size.

4.2.2 Stacking

To combine models with the goal of minimizing the error rate, Wolpert (1992) proposed the idea of stacking. It is similar to cross validation in that it chooses a model, however it is not limited to choosing the best existing model. Stacking is the combining of predictive models, mapping the covariates ($\boldsymbol{X}$) to an outcome ($\mathbf{Y}$). These models, or learners, are built on a learning set. This concept is brought into the statistcal setting through stacked regression (Breiman, 1996). The mean squared prediction error (MSPE) of the combined predictors from the k learners ($\hat{y}_i^k$) can be minimized through regression on the coefficients α_k.

$$MSPE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{k=1}^{K}\alpha_k \hat{y}_i^k\right)^2 \tag{4.2}$$

By combining models built from a learning set, we can use a training set to combine the models in a way that further minimizes the error rate from that of the best single model, as in cross validation. This is done by taking the prediction from the learners and using them as covariates to perform prediction on a test data set. Wolpert (1992) mentions that if the training set model stated that you simply choose the predictors most correlated with the output, this would be equivalent to choosing the best model from the learning set, which would be exactly cross-validation. Rather, if the training set model was some weighted average of the inputs, this would tell us to combine our learners with the given weights. This is exactly what is done with stacked regression described above.

Wolpert (1992) described a set of simulations with numerical data. Data were generated from polynomial or trigonometric functions. Learners were fit on a learning set. The best of these learners was compared to a stacked model. The stacking approach decreased the error by a factor of 2 in the trigonometric and third-order polynomial case, and decreased error by a factor of 50 in the second-order polynomial case.

Since regression typically overfits the data, leave-one-out cross validation is suggested to provide better generalization by Breiman (1996). When data sets are large, this creates a computational burden; 10 fold cross validation is then suggested. However, since the predicting models are correlated, this can still be a difficult problem. Since the goal is to improve prediction, each α_k from (4.2) is constrained to be non-negative. Breiman also suggests using Ridge Regression (Hoerl and Kennard, 1970) to constrain

$\sum |\alpha_k|$ to account for the correlation. This then also leads to few learners to be included in the combined predictor, reducing overfitting.

In simulations of Breiman's method, he found that using stacked regression never provides worse results and often produces models with less prediction error than the "best" of the learners. However, Džeroski and Ženko (2004) claims that many stacking approaches provide similar results to that of the best learner. Specific advances in stacking have come in the forms of SCANN (Stacking Correspondence Analysis, Nearest Neighbor) (Merz et al., 1999), meta-decision trees (Todorovski, 2003), stacking with probability distributions and multi-response linear regression (Džeroski and Ženko, 2004). Džeroski and Ženko (2004) found that in their particular simulations, their methods performed better than other stacking methods. This may have been due to types of simulations conducted. They acknowledged their method works better when the learners considered were from different families rather than adjusted parameters among the same learning algorithms.

4.2.3 Super Learner

A way to perform stacking utilizing a statistical rule is Super Learner (Van Der Laan et al., 2007). Super Learner combines many different learners, or algorithms, into a single model. This is done in exactly the same manner as stacked regression in equation (4.2). There are times when we aren't sure which of several model classes is appropriate for a problem. Super Learner allows us to try many learners and combine them in an efficient way in order to perform prediction.

We can use Super Learner to minimize the squared error loss function using cross-validation. This is done by performing least squares regression a training set on the predicted values from several models that have been fit from a learning set.

$$\hat{y}_i = \sum_{k=1}^{K} \alpha_k \hat{y}_i^k \tag{4.3}$$

Super Learner has the oracle property; asymptotically, for the best model k, α_k will converge to 1, while for other $j \neq k$, α_j converges to 0. Super Learner will choose a model that performs no worse than the correct model for the true data generating mechanism.

In an R package, Super Learner has combined models produced by the learners Least Angle Regression (Efron et al., 2004), Logic Regression (Ruczinski et al., 2003), D/S/A (Sinisi and van der Laan, 2004), Regression Trees (Therneau et al., 1997), Ridge Regression (Hoerl and Kennard, 1970), Random Forests (Liaw and Wiener, 2002), and Adaptive Regression Splines (Friedman and Roosen, 1995).

By plugging in the predictions of our K models and our $\hat{\alpha_k}$'s into our Super Learner fit on a test set, we should be able to improve or match the prediction from the best model that we fit. This is an easy way to use the concept of stacking to combine models to perform prediction.

The goal of Super Learner is to determine the best approach to obtaining a prediction model when the data generating mechanism is unknown. It has been applied to data that is relatively clean, but when it is unclear which modeling approach is optimal. We plan to leverage this idea of combining models determined through different approaches, however, we wish to use it on models that differ in the form of how they correct for a source of potential bias in the data.

4.3 Methods

To perform variable selection in the presence of multiple data issues, we utilize principles from machine learning along with methods to deal with data issues in the low-dimension setting. We incorporate the corrections into a boosting algorithm which generates a variable selection path, i.e., a sequence of vectors of regression model coefficient estimates, where many of the coefficients are zero. Further, we utilize stacking to combine these estimates together to provide even more candidate models to choose from, possibly improving the prediction accuracy of our model. Lastly, from the candidates, we use cross validation to select the most accurate model and simultaneously perform variable selection.

4.3.1 Boosting

Boosting is used to improve the prediction of an existing model (Schapire, 2003). We can create an iterative procedure that builds on the model from the previous step from the initial null model. This creates a path of solutions that we can choose from using cross validation. The most simple example of boosting that we will build upon comes from gradient descent (Friedman et al., 2000).

This method was generalized by Wolfson (2011) to accommodate a breach in the standard regression assumptions such as correlated outcomes. Suppose our goal is to perform variable selection in the regression model defined by

$$\hat{\mathbf{Y}} = \beta_0 + \sum_{j=1}^{J} \boldsymbol{X}'_j \beta_j \tag{4.4}$$

We replace the gradient of the log likelihood with a set of estimating equations,

$\mathbf{g}(\boldsymbol{x}, \boldsymbol{\beta})$, that behaves in a similar way. Brown et al. (2017) added a thresholding component in ThrEEBoost (Thresholded Boosting) to consider a larger set of possible solutions. The general form of this algorithm contains three parameters: T: Number of iterations to perform, ϵ, the step-size to increase $\hat{\beta}_i$, and τ, the threshold that controls how many coefficients are updated at each step.

Algorithm 7 ThrEEBoost

```
procedure ThrEEBoost
Set $\boldsymbol{\beta}^{(0)} = \mathbf{0}$
    for $t = 0, \ldots, T$ do
        Compute $\boldsymbol{\Delta} = \mathbf{g}(\mathbf{X}, \boldsymbol{\beta})_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(t-1)}}$
        Identify $J_t = \{j : |\boldsymbol{\Delta}_j| \geq \tau \cdot \max_j |\boldsymbol{\Delta}_j|\}$
        for all $j_t \in J_t$ do
            Update $\boldsymbol{\beta}_{j_t}^{(t)} = \boldsymbol{\beta}_{j_t}^{(t-1)} + \epsilon \text{ sign}(\Delta_{j_t})$
```

To accomodate correlated outcomes, we utilize GEE where with link function $h()$ such that the mean of y, $\boldsymbol{\mu}_i = h^{-1}(\boldsymbol{X}_i \boldsymbol{\beta})$. We then use the following function in ThrEEBoost:

$$\mathbf{g}(\mathbf{X}, \boldsymbol{\beta}) = \sum_{i=1}^{n} \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}} \mathbf{V}_i^{-1} (\mathbf{Y}_i - \boldsymbol{\mu}_i). \tag{4.5}$$

For measurement error, we can utilize the corrected score functions from Nakamura (1990) as was done in MEBoost (Brown et al., 2017). Here we use the following function in the algorithm above and simultaneously estimate the variance of the outcome where $S(\mathbf{Y}, \mathbf{X}, \beta)$ is the naive score function assuming $\mathbf{X}$ is measured correctly, and Σ_{UU} is the variance of the additive measurement error term.

$$\mathbf{g}(\mathbf{X}, \boldsymbol{\beta}) = S(\mathbf{Y}, \mathbf{X}, \beta)' + n\sigma^{-2}\beta' \Sigma_{UU} \tag{4.6}$$

In each case, the form of the iterative procedure can be altered to deal with one a single violation of a regression assumption. We wish to be able to account for multiple violations by combining these algorithms together.

4.3.2 Combining the methods

While there are stand-alone methods to address a single violation of our standard assumptions, our options are limited when two or more are violated. We describe below our proposed method in the setting where outcomes are correlated and covariates are measured with error, though we emphasize that the framework could be applied to account for more than two types of violations of standard assumptions. Since both GEEBoost and MEBoost are iterative procedures with many steps, we can try to combine the approach at each iteration to create a unique coefficient path, directed by the multiple $\mathbf{g}_k(\mathbf{X}, \boldsymbol{\beta})$ functions. Another option is to combine the full coefficient paths, $\boldsymbol{\beta}_k$ created by each learner afterwards using a stacking approach, such as Super Learner.

At each iteration of the boosting procedure above, we incorporate a rule to update coefficients based on the functions from the different boosting procedures denoted k the algorithm, $\mathbf{g}_k(\mathbf{X}, \boldsymbol{\beta})$. Some of the possibilities are:

1. $\mathbf{g}^*(\mathbf{X}, \boldsymbol{\beta}) = \sum w_k \mathbf{g}_k(\mathbf{X}, \boldsymbol{\beta})$
2. $\mathbf{g}^*(\mathbf{X}, \boldsymbol{\beta}) = \prod \mathbf{g}_k(\mathbf{X}, \boldsymbol{\beta})$
3. Updating $\boldsymbol{\beta}_j$ only if thresholds are met in both estimating equations.
4. Cycle through $\mathbf{g}^*(\mathbf{X}, \boldsymbol{\beta}) = \mathbf{g}_k(\mathbf{X}, \boldsymbol{\beta})$ so that a different $\mathbf{g}_k(\mathbf{X}, \boldsymbol{\beta})$ is used for the next iteration.

As another option in this paper, we will consider the 4th possibility listed. This will be referred to as AltBoost (Alternating Estimating Equation Boost). In a preliminary set of simulations, we found that this method performed the best of the 4 approaches listed above with varied weights.

4.3.3 SuperBoost

Another approach is to combine the paths produced by each boosting procedure used. The boosting procedures can vary in the parameters used or the function $\mathbf{g}(\mathbf{X}, \boldsymbol{\beta})$. Utilizing the practice of stacking, we can bring these estimates together to improve the prediction of our model. One statistical tool we can use to do this is Super Learner (Van Der Laan et al., 2007). Since our goal is to select a point from a path so that we can limit the number of variables included, we still wish to combine $\boldsymbol{\beta}$ of a similar magnitude.

SuperBoost (Super Learner Estimating Equations Boost), utilizes Super Learner (Van Der Laan et al., 2007) to combine models to improve prediction. While Super Learner generally is used to combine different learning approaches (e.g. Least Angle Regression (Efron et al., 2004) and Random Forests (Liaw and Wiener, 2002)), we can use learners hat address different violations to the standard regression assumptions. By combining these together, we may be able to pick up signals that violate certain assumptions in each learner.

For example, if we have correlated outcomes y_{ij} and mismeasured covariates ($\boldsymbol{X}'_{ij}$), we can utilize corrections for each of these two assumption violations that correct for each individually. Let $\boldsymbol{B}_G$ and $\boldsymbol{B}_M$ be the variable selection paths obtained by applying GEEBoost and MEBoost respectively to our data. Further, for some magnitude of the coefficients $l = ||\boldsymbol{\beta}||_1$, and the corresponding coefficients $\boldsymbol{\beta}_G$ and $\boldsymbol{\beta}_M$ we use Super Learner to predict outcomes y_{ij} in our test set by combining the learners created by these two approaches through estimated model weights, w^l_G and w^l_M for GEEBoost and MEBoost, respectively. We can predict $\hat{y_{ij}} = w^l_G \boldsymbol{X}'_{ij} \boldsymbol{\beta}^l_G + w^l_M \boldsymbol{X}'_{ij} \boldsymbol{\beta}^l_M$. This may allow us to improve the prediction over each individual learner, $\boldsymbol{\beta}^l_G$ and $\boldsymbol{\beta}^l_M$.

Further, each of these learners can be varied through the selection of parameters needed for each algorithm. For example, we can vary the threshold τ used in each algorithm to create even more candidate coefficients. We may have a unique path $\boldsymbol{\beta}_G^{\tau}$ for each threshold used within GEEBoost. We can combine this larger number of learners to improve our prediction further.

The SuperBoost algorithm incorporates multiple threshold values in multiple boosting algorithms to create K unique learners, or selection paths. From these paths, we perform Super Learner for a subset of $l = ||\boldsymbol{\beta}||_1$. For each value of l, we fit a regression model using the predictions from each learner ($\hat{y}_{ij}^l$) as the predictors to obtain pseudo-weights. The predictions are combined with the adjusted (non-negative and sum to 1) weights. This creates the SuperBoost path of solutions which we can select from using cross-validation.

Considering variable selection, at any point on our path, if one of the learners has selected the variable and that learner has a non-zero weight, it would also be selected into the model. This could rapidly deflate the specificity of the model when the number of learners considered (K) becomes large. Therefore, another parameter ζ can be added to control the specificity of the SuperBoost method. Since predictor p is selected whenever $\hat{\beta}_p^* \neq 0$, we propose to set $\hat{\beta}_p^* = 0$ if $|\hat{\beta}_p^*| < \zeta$. Selecting small value of ζ as a threshold into the model will help us preserve the specificity by making it more difficult for a variable to be included by chance.

The parameters for SuperBoost are:

- K = Number of learners to combine
- L = Maximum magnitude of $\hat{\beta}^*$ to be considered
- ζ = Threshold of $|\hat{\beta}_p^*|$ to included variable p into the model

Algorithm 8 SuperBoost

```
procedure SuperBoost
    for $k = 1, \ldots, K$ do
        Perform a boosting algorithm to produce a path of coefficients, $\hat{\boldsymbol{B}}_k$ on a learning set $(\mathbf{Y}_r, \boldsymbol{X}_r)$
    for $l = ||\hat{\boldsymbol{\beta}}||_1$ in (0, L): do
        1. Let $\hat{\boldsymbol{\beta}}_k^l \in \hat{\boldsymbol{B}}_k$ with $||\hat{\boldsymbol{\beta}}_k^l||_1 = l$
        2. Use a training set of data $(\mathbf{Y}_s, \boldsymbol{X}_s)$, use regression to obtain pseudo-weights $(\alpha_k)$ of each learner:
            $$E(\mathbf{Y}_s) = \alpha_0 + \sum_{k=1}^{K} \alpha_k \boldsymbol{X}_s \hat{\boldsymbol{\beta}}_k^l$$
        .
        3. Obtain the weights $(w_k)$ by scaling the pseudo-weights $(\hat{\alpha_k})$ so that they are positive and sum to 1:
            $$w_k = max(0, \hat{\alpha_k}) / \sum_{k=1}^{K} max(0, \hat{\alpha_k}).$$
        4. Obtain $l$-th SuperBoost estimate:
            $$\hat{\boldsymbol{\beta}}_{SB}^l = \sum_{k=1}^{K} w_k \hat{\boldsymbol{\beta}}_k^l$$
        5. If $|\hat{\boldsymbol{\beta}}_{SB}^l| < \zeta$, then set $\hat{\boldsymbol{\beta}}_{SB}^l$=0.
        6. Obtain the mean squared error (MSE) of the solution $\hat{\boldsymbol{\beta}}_{SB}^l$ from a test set $(\mathbf{Y}_t, \boldsymbol{X}_t)$
    From new SuperBoost path $\boldsymbol{B}_{SB}$, Select $l*$ such that $\hat{\boldsymbol{\beta}}_{SB}^{l*}$ that minimizes MSE on the test set, $(\mathbf{Y}_t, \boldsymbol{X}_t)$.
```

4.4 Simulation

4.4.1 Set-up

We generated data under three different scenarios to assess the performance of Super-Boost. In the first scenario, outcomes were correlated and covariates were measured with error, and these two factors had a similar degree of influence on the variable selection performance of the models we considered. In the second scenario, correlation had a larger influence than measurement error, while in the third scenario measurement error was more influential than correlation. We varied the number of clusters, n, the variance of the outcome, σ^2, the measurement error, δ, the pairwise correlation in outcomes within a cluster, ρ, and the effect sizes β_m and β_g for covariates measured with and without error, respectively. The values of these parameters for each scenario are listed in table 4.1.

Scenario	Description	n	σ^2	δ	ρ	β_m	β_g
1a	Correlation and ME have similar influence	40	1	1.0	0.6	0.4	0.8
1b	- less information	30	2	1.0	0.6	0.2	0.3
2a	Correlation more influential	40	1	0.2	0.7	0.2	0.3
2b	- less information	30	2	0.2	0.7	0.3	0.4
3a	Measurement Error more influential	40	1	1.0	0.2	0.6	0.5
3b	- less information	30	2	1.0	0.2	0.3	0.2

Table 4.1: List of parameter values for each scenario. n is the number of clusters, σ^2 is the residual variance, δ is the standard deviation of the measurement error, ρ is the pairwise correlation in outcomes within clusters, β_m is the effect size of mismeasured predictors, β_g is the effect size of predictors measured without error. Scenario b in each setting has fewer clusters and higher residual variance, giving us less information to reach an accurate estimate.

Data were generated with 100 covariates coming from a multivariate normal distribution. In clusters of 10 covariates within each observation, there was pairwise correlation of $\phi = 0.3$. Of the 100 predictors, 16 had truly non-zero coefficients. Measurement error occurred in 50 of the covariates with standard deviation δ. Outcomes were generated for n clusters of 6 observations from a multivariate normal distribution. Within clusters, outcomes have a pairwise-correlation of ρ. The residual error was normally distributed with variance σ^2. We conducted 500 simulations for each algorithm and each scenario. We compared the existing methods of GEEBoost, MEBoost, CoCoLasso, and Lasso, with our proposed methods of AltBoost and SuperBoost. Each of the boosting methods were performed with threshold values of of $\tau = 0.4, 0.6$, and 0.8.

We used six metrics to assess the performance of our methods. First, mean squared error (MSE) was used to quantify the prediction error of the final selected model. Next, the L_1 distance from the true parameters, computed as $||\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}||_1$, was used to quantify the bias. We also computed variable selection sensitivity (the portion of variables correctly included in the model) and specificity (the portion of variables correctly excluded from the model). Next, the mean range of candidate $||\hat{\boldsymbol{\beta}}||_1$ values that are within 5% of the minimum MSE for that simulation. This gives us an idea of the range of chosen $l = ||\hat{\boldsymbol{\beta}}||_1$ that will perform well. Lastly the standard deviation of the L_1 magnitude of the estimate across the simulations, $SD(||\hat{\boldsymbol{\beta}}||_1)$. This tells us how wide the range of selected $l = ||\hat{\boldsymbol{\beta}}||_1$ was, which indicates how important it is to accurately choose $||\hat{\boldsymbol{\beta}}||_1$.

4.4.2 Results

In scenario 1a, we found that two of our newly proposed methods each obtained the same minimum value for MSE, SuperBoost and AltBoost with a threshold of 0.8. Utilizing SuperBoost improved our MSE by 7.9% over the previous best methods of MEBoost with a threshold of 0.8 and GEEBoost with a threshold of 0.6. SuperBoost also had a similar or higher sensitivity (1.00) and specificity (0.84) then those two models. AltBoost saw a larger gain in specificity (0.86). In the more difficult scenario 1b, SuperBoost improved prediction by 2.6% over the best standalone learner, MEBoost with $\tau = 0.6$.

The best method in terms of MSE was GEEBoost combined with SuperBoost. The MSE was 3.0% better than MEBoost with $\tau = 0.6$. When we compare these new methods with the best existing method, we gained in specificity (both 0.88 vs. 0.73) without sacrificing and sensitivity (0.93 for SuperBoost and MEBoost - 0.6, and 0.94 for GEEBoost - Super).

	MSE	$\|\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|\|_1$	Sens.	Spec.	Range	$SD(\|\|\hat{\boldsymbol{\beta}}\|\|_1)$
AltBoost - 0.4	1.59	4.14	1.00	0.69	1.25	0.51
AltBoost - 0.6	1.43	3.02	1.00	0.80	1.31	0.57
AltBoost - 0.8	1.40	2.66	1.00	0.86	1.22	0.55
MEBoost - 0.4	1.67	4.84	1.00	0.51	1.46	0.51
MEBoost - 0.6	1.54	3.67	1.00	0.65	2.10	0.72
MEBoost - 0.8	1.52	3.28	1.00	0.73	2.16	0.72
MEBoost - Super	1.54	3.12	0.97	0.90	1.57	0.43
GEEBoost - 0.4	1.80	5.05	0.98	0.65	1.71	0.51
GEEBoost - 0.6	1.52	3.37	0.98	0.81	1.80	0.47
GEEBoost - 0.8	1.54	2.97	0.97	0.88	1.55	0.46
GEEBoost - Super	1.47	2.95	1.00	0.83	1.79	0.54
Lasso	1.59	3.39	1.00	0.63	2.92	0.58
CoCoLasso	1.77	3.18	0.95	0.91	3.01	0.39
SuperBoost	1.40	2.88	1.00	0.84	1.54	0.51

Table 4.2: Performance metrics for scenario 1a.

In scenario 2a, where correlation was the more pronounced issue than measurement error, SuperBoost performed slightly better than model using the optimal threshold (0.6) from GEEBoost. MSE was 4.3% better and specificity was much better (0.93 vs 0.82) while sensitivity remained at 1. However, by combining the models of different thresholds using the GEEBoost algorithm, utilizing SuperBoost allowed us to improve MSE by 6.1% while specificity increased to 0.93 from 0.82. Scenario 2b provided similar results; MSE was decreased by 5.3% combining GEEBoost algorithms using Super Learner, while increasing sensitivity (0.97 vs. 0.92) and specificity (0.92 vs 0.81).

In scenario 3a, where measurement error was the primary issue, we saw that using the SuperBoost algorithm using only the MEBoost models as components, helped us

	MSE	$\|\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|\|_1$	Sens.	Spec.	Range	$SD(\|\|\hat{\boldsymbol{\beta}}\|\|_1)$
AltBoost - 0.4	2.35	3.09	0.97	0.60	1.18	0.33
AltBoost - 0.6	2.27	2.34	0.95	0.76	1.57	0.42
AltBoost - 0.8	2.33	2.49	0.91	0.84	1.59	0.49
MEBoost - 0.4	2.38	3.19	0.96	0.63	1.23	0.35
MEBoost - 0.6	2.35	2.67	0.93	0.73	1.68	0.48
MEBoost - 0.8	2.40	2.86	0.90	0.78	1.71	0.53
MEBoost - Super	2.45	2.67	0.86	0.88	1.39	0.41
GEEBoost - 0.4	2.40	3.11	0.93	0.71	1.25	0.32
GEEBoost - 0.6	2.38	2.67	0.85	0.83	1.70	0.50
GEEBoost - 0.8	2.48	2.81	0.79	0.89	1.71	0.50
GEEBoost - Super	2.28	2.04	0.94	0.88	1.10	0.32
Lasso	2.46	3.04	0.92	0.71	2.78	0.61
CoCoLasso	2.64	3.04	0.76	0.90	2.06	0.47
SuperBoost	2.29	2.11	0.93	0.88	1.07	0.32

Table 4.3: Performance metrics for scenario 1b.

improve MSE by 22.1%. The L_1 distance from the true solution was reduced by 29.3% and specificity was increased from 0.64 to 0.87. Scenario 3b also showed an decrease in prediction error, but not as drastic (1.2%). Though the L_1 distance from the true $\boldsymbol{\beta}$ was decreased by 12.0%.

The weights and traceplots of the coefficients are shown for scenario 1a in Figure 4.1. The first row shows the SuperBoost algorithm that combines 3 thresholds of MEBoost and 3 thresholds of GEEBoost. In the second row, we show how the weights and coefficients behave if we only consider the 3 MEBoost thresholds. In the last row, we show how the weights and coefficients behave if we only consider the 3 GEEBoost thresholds. As the magnitude of the coefficients grows, more weight is placed on the algorithms that use a larger threshold. Early in the iterative process, many variables are included into the model. However, as the coefficients get larger,we become more strict about which values get updated. This is demonstrated for scenario 3a in Figure 4.1.

	MSE	$\|\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|\|_1$	Sens.	Spec.	Range	$SD(\|\|\hat{\boldsymbol{\beta}}\|\|_1)$
AltBoost - 0.4	1.18	2.41	1.00	0.62	1.35	0.38
AltBoost - 0.6	1.11	1.61	1.00	0.76	1.39	0.39
AltBoost - 0.8	1.10	1.59	1.00	0.85	1.42	0.40
MEBoost - 0.4	1.22	2.47	1.00	0.68	1.36	0.40
MEBoost - 0.6	1.16	1.76	1.00	0.79	1.52	0.40
MEBoost - 0.8	1.15	1.80	1.00	0.85	1.50	0.43
MEBoost - Super	1.14	1.60	1.00	0.92	1.13	0.31
GEEBoost - 0.4	1.20	2.36	1.00	0.71	1.36	0.40
GEEBoost - 0.6	1.15	1.72	1.00	0.82	1.48	0.39
GEEBoost - 0.8	1.16	1.75	1.00	0.87	1.45	0.40
SuperBoost	1.08	1.21	1.00	0.93	0.85	0.23
Lasso	1.20	1.90	1.00	0.76	2.23	0.39
CoCoLasso	1.20	1.82	1.00	0.84	5.20	0.33
SuperBoost	1.10	1.22	1.00	0.93	0.83	0.24

Table 4.4: Performance metrics for scenario 2a.

The range and standard deviation of the L_1 magnitude of the selected point estimate were not smaller for most cases using SuperBoost vs. a standalone learning procedure. This is contrary to what we expected. Combining the methods together helped us minimize the MSE further, however, the minimum was reached rapidly and did not provide us a wider range for a value of $||\hat{\boldsymbol{\beta}}||_1$ that would allow us to perform prediction as well. For scenario 3a, this is shown in Figure 4.2 for MEBoost with specified thresholds vs MEBoost where we combined paths using SuperBoost. Since the weights change as the path progresses with $||\hat{\boldsymbol{\beta}}||_1$, we expected the MSE to stay at a low point for longer, however this was not the case. Since the MSE dropped to a lower point from the same starting point, the curve had to be steeper to reach the minimum.

	MSE	$\|\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|\|_1$	Sens.	Spec.	Range	$SD(\|\|\hat{\boldsymbol{\beta}}\|\|_1)$
AltBoost - 0.4	2.22	2.53	0.99	0.59	1.30	0.30
AltBoost - 0.6	2.15	1.93	0.98	0.76	1.50	0.40
AltBoost - 0.8	2.18	2.01	0.96	0.84	1.59	0.44
MEBoost - 0.4	2.29	2.62	0.97	0.71	1.37	0.39
MEBoost - 0.6	2.25	2.25	0.94	0.80	1.69	0.47
MEBoost - 0.8	2.29	2.44	0.91	0.84	1.67	0.49
MEBoost - Super	2.31	2.22	0.90	0.89	1.33	0.37
GEEBoost - 0.4	2.30	2.62	0.96	0.72	1.38	0.39
GEEBoost - 0.6	2.26	2.21	0.92	0.81	1.67	0.46
GEEBoost - 0.8	2.35	2.50	0.88	0.86	1.72	0.48
GEEBoost - Super	2.14	1.47	0.97	0.92	1.01	0.28
Lasso	2.37	2.69	0.92	0.79	2.65	0.60
CoCoLasso	2.40	2.70	0.87	0.85	3.59	0.55
SuperBoost	2.17	1.57	0.97	0.92	0.98	0.28

Table 4.5: Performance metrics for scenario 2b.

4.5 Discussion

We introduced two methods, AltBoost and SuperBoost, to perform variable selection in the presence of multiple data issues. These are just a couple possible ways to combine methods to address this unanswered problem. SuperBoost is a general purpose method that can combine the results from multiple boosting techniques. The existing ThrEEBoost framework provides a simple approach to implementing a boosting technique which accounts for one particular data challenge, and SuperBoost gives the recipe for combining them. SuperBoost can also utilize any learning procedure that produces a path of coefficients.

As in the general idea of stacking, we can improve our prediction by combining estimators rather than choosing the best with cross validation. In ThrEEBoost when we choose the best single threshold, we are able to predict well sometimes, but using SuperBoost allows us to improve prediction by combining the predictions from the paths produced by different thresholds. SuperBoost is appealing because it is an "all-in-one"

	MSE	$\|\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|\|_1$	Sens.	Spec.	Range	$SD(\|\|\hat{\boldsymbol{\beta}}\|\|_1)$
AltBoost - 0.4	1.93	4.88	1.00	0.59	1.57	0.61
AltBoost - 0.6	1.69	3.68	1.00	0.76	1.25	0.67
AltBoost - 0.8	1.74	3.51	0.99	0.86	1.12	0.64
MEBoost - 0.4	1.99	5.11	1.00	0.52	1.96	0.67
MEBoost - 0.6	1.95	4.40	1.00	0.64	2.25	0.75
MEBoost - 0.8	1.98	4.44	1.00	0.69	2.21	0.77
MEBoost - Super	1.52	3.11	1.00	0.87	1.68	0.50
GEEBoost - 0.4	1.53	3.68	1.00	0.73	1.77	0.56
GEEBoost - 0.6	1.58	3.18	1.00	0.83	1.76	0.53
GEEBoost - 0.8	1.88	3.72	0.98	0.88	1.64	0.55
GEEBoost - Super	1.97	4.19	1.00	0.78	2.06	0.62
Lasso	2.13	4.66	1.00	0.61	3.45	0.71
CoCoLasso	2.09	3.84	0.98	0.91	2.25	0.47
SuperBoost	1.62	3.48	1.00	0.83	1.54	0.60

Table 4.6: Performance metrics for scenario 3a.

approach, not requiring cross-validation at the end to choose a threshold.

Through a simulation study, we saw that using SuperBoost performed as well as the standalone methods to address one issue, and performed better when the two issues we explored (correlation and measurement error) had similar effects on the difficulty of the estimation. In the scenarios where measurement error and correlation affected prediction in a similar magnitudes, SuperBoost and AltBoost performed better then the standalone methods in terms of MSE. Using these methods improved prediction error by 7.9% each in scenario 1a, while SuperBoost and AltBoost improved MSE by 2.6% and 4.3% over existing methods, respectively.

In the other two scenarios, SuperBoost did nearly as well as the method that corrected for the issue that had a larger impact on the misspecification of the standard model. We also found that using SuperBoost only considering the correction for a single data challenge allowed us to improve prediction by combining the learners rather than selecting the best with cross-validation. For the second set of scenarios, where correlation was more pronounced, 2a and 2b, we improved our prediction error by 4.3%

	MSE	$\|\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|\|_1$	Sens.	Spec.	Range	$SD(\|\|\hat{\boldsymbol{\beta}}\|\|_1)$
AltBoost - 0.4	2.48	3.26	0.97	0.59	1.29	0.37
AltBoost - 0.6	2.44	3.01	0.93	0.72	1.60	0.44
AltBoost - 0.8	2.53	3.08	0.85	0.83	1.70	0.53
MEBoost - 0.4	2.50	3.29	0.95	0.63	1.47	0.42
MEBoost - 0.6	2.50	3.09	0.92	0.71	1.74	0.52
MEBoost - 0.8	2.52	3.20	0.88	0.77	1.80	0.59
MEBoost - Super	2.47	2.72	0.87	0.88	1.41	0.46
GEEBoost - 0.4	2.42	2.88	0.94	0.75	1.42	0.42
GEEBoost - 0.6	2.46	2.79	0.85	0.84	1.77	0.49
GEEBoost - 0.8	2.64	3.13	0.75	0.90	1.79	0.51
GEEBoost - Super	2.54	3.22	1.00	0.82	2.21	0.62
Lasso	2.62	3.50	0.89	0.69	2.95	0.69
CoCoLasso	2.76	3.37	0.72	0.91	2.22	0.54
SuperBoost	2.49	3.25	1.00	0.82	1.97	0.57

Table 4.7: Performance metrics for scenario 3b.

and 5.3%. When measurement error was the more difficult problem, we reduced prediction error by 22.1% and 1.2% by combining the paths created by different thresholds of MEBoost.

We also noticed with the SuperBoost method that when $||\hat{\beta}||_1$ was small, more weight is placed on the algorithms with lower thresholds, which includes more variables. Later in the path, high thresholds have higher weights to control specificity. Contrary to our expectation, we did not see a gain in the width of the $||\hat{\beta}^*||_1$ that allowed us to choose a point on the path that was “close” to minimizing the MSE on the test set. This was shown through the standard deviation of the selected point and the range of points within 5% of the minimum MSE value.

This method was shown in the case of correcting data with correlated outcomes and additive measurement error. Utilizing other EEBoost algorithms, this method could be used in other cases of assumption violations included multiplicative measurement error, missing data, and censored data.

Further work could be performed on a wider variety of assumption violations and simulation scenarios. This method is a first step in addressing issues that arise when accounting for multiple sources of bias in the estimation of a model. We proposed two methods and showed that they worked well in a simulation, however we hope that we can show that the method generalizes to a wider variety of data and has applications in the real world.

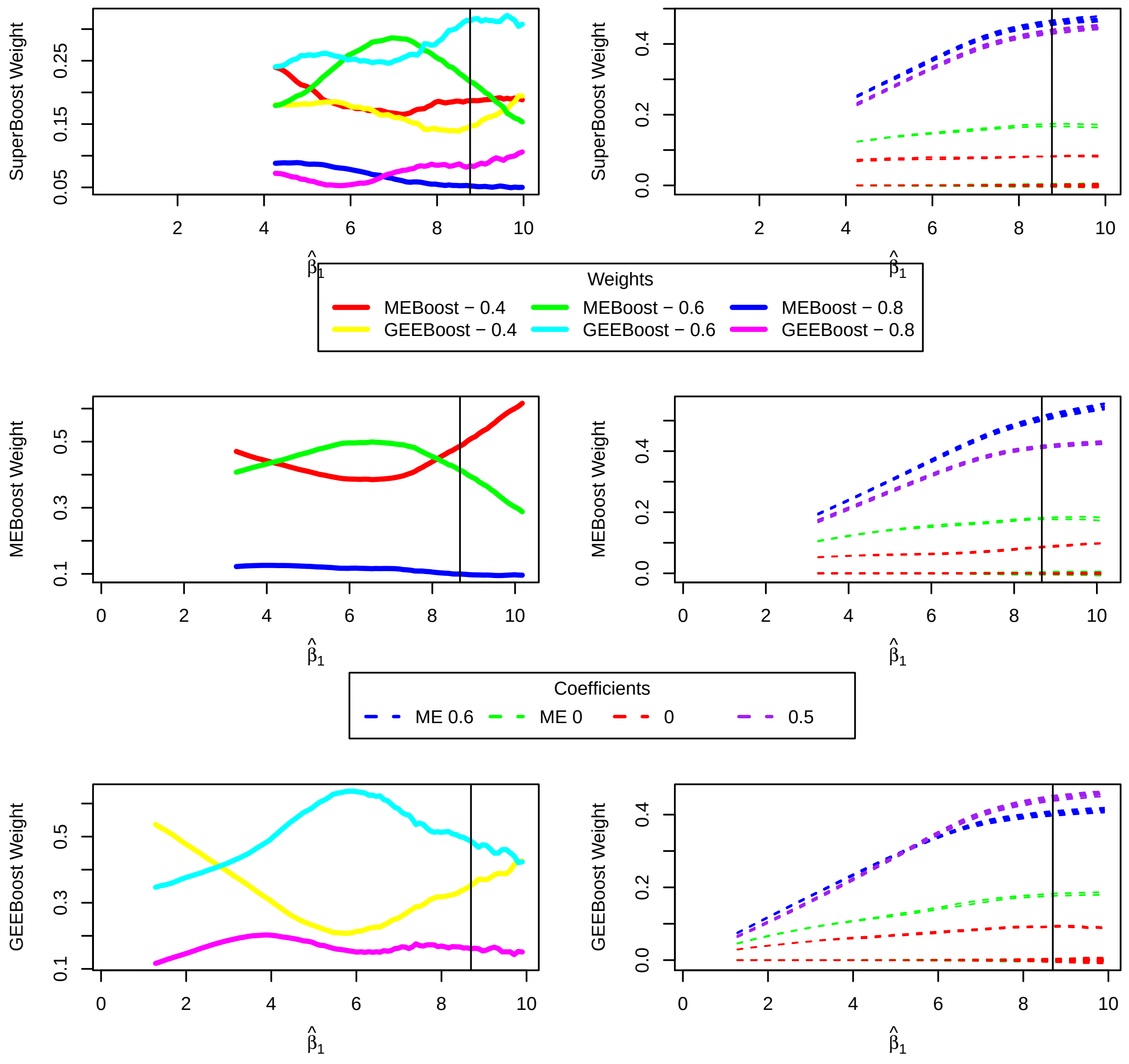

Figure 4.1: Model weights and traceplots for mean paths of Super Learner models in scenario 1b.

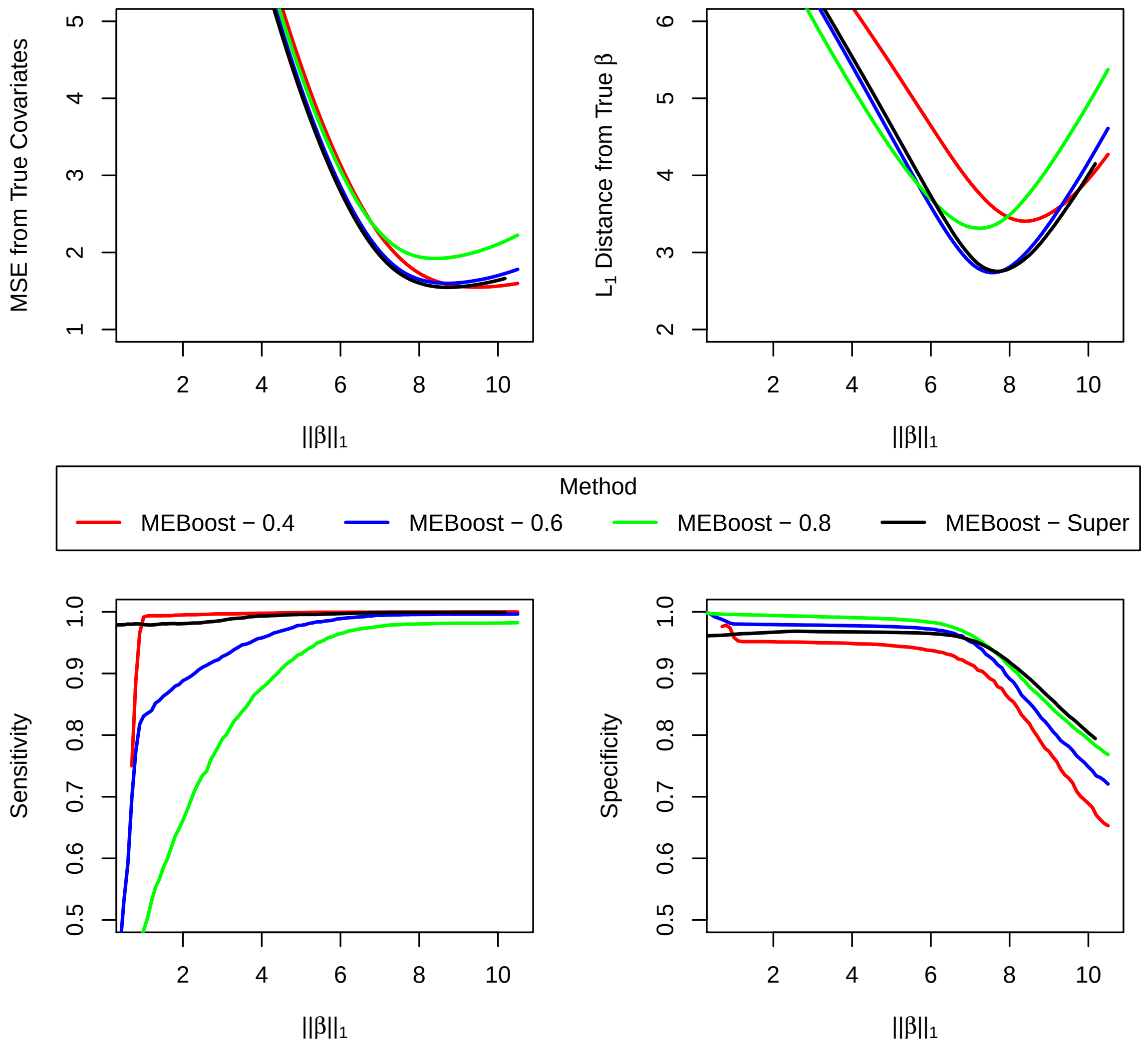

Figure 4.2: Performance metrics over the paths created by MEBoost in scenario 3a.

Chapter 5

Conclusion and Discussion

The method ThrEEBoost, allows us to update multiple coefficients at each iteration through a threshold parameter. By allowing more coefficients to be updated at each iteration, ThrEEBoost can explore a greater diversity of variable selection "paths" through the model space. Through a simulation, we showed that the method performed as well as EEBoost for very sparse models, and improved prediction error in a less sparse scenario than any of those on the path defined by EEBoost. We also showed in Box Lunch study that we were able to improve the prediction of BMI using ThrEEBoost rather than EEBoost or the LASSO.

To perform variable selection and prediction when covariates are measured with error, we used MEBoost. In our simulation study, we showed that for many sources of additive measurement error, using MEBoost can improve prediction error over the CoCoLasso or LASSO. However, the CoCoLasso had higher specificity than MEBoost in all cases. When we applied our method to the Box Lunch Study, we found that MEBoost was more accurate at predicting the count data, number of times a subject binged in the last month.

To analyze a single data set that encounters more than one data challenge, we proposed SuperBoost. This method can combines models together to improve the prediction over the best one. In creating the SuperBoost path for correlated outcomes whose covariates are measured with error, we found that our method performed no worse than the best single learning model that accounted for only one of the two issues. In some scenarios, using SuperBoost improved prediction. This method is meant to address a problem in an area where not much research has been conducted and provides a first step into thinking about how to address very “messy” data.

References

Breiman, L. (1996). Stacked Regressions. *Machine Learning* **24,** 49–64.

Brown, B., Miller, C. J., and Wolfson, J. (2017). ThrEEBoost: Thresholded Boosting for Variable Selection and Prediction via Estimating Equations. *Journal of Computational and Graphical Statistics* pages 1–10.

Brown, B., Weaver, T., and Wolfson, J. (2017). MEBoost: Variable Selection in the Presence of Measurement Error. *ArXiV.org* .

Bunea, F., Tsybakov, A., and Wegkamp, M. (2007). Sparsity oracle inequalities for the Lasso. *Electronic Journal of Statistics* **1,** 169–194.

Buonaccorsi, J. (2010). *Measurement Error: Models, Methods and Applications.* CRC Press, Boca Raton.

Buonaccorsi, J., Demidenko, E., and Tosteson, T. (2000). Estimation in Longitudinal Random Effects Models with Measurement Error. *Statistica Sinica* **10,** 885–903.

Buonaccorsi, J. P. (1995). Prediction in the Presence of Measurement Error: General Discussion and an Example Prediction in the Presence of Measurement Error: General Discussion and an Example Predicting Defoliation. *Source: Biometrics* **51,** 1562–1569.

Cefalu, M., Dominici, F., Arvold, N. D., and Parmigiani, G. (2016). Model Averaged Double Robust Estimation. *Biometrics* .

Datta, A. and Zou, H. (2017). Cocolasso for high-dimensional error-in-variables regression. *Annals of Statistics* .

Díaz, I., Hubbard, A., Decker, A., and Cohen, M. (2015). Variable importance and prediction methods for longitudinal problems with missing variables. *PloS one* **10,** e0120031.

Džeroski, S. and Ženko, B. (2004). Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning* **54,** 255–273.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least Angle Regression. *The Annals of Statistics* **32,** 407–451.

Fan, J. and Li, R. (2001). Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. *Journal of the American Statistical Association* **96,** 1348–1360.

Fan, J. and Li, R. (2002). Variable Selection for Cox's Proportional Hazards Model and Frailty Model. *The Annals of Statistics* **30,** 74–99.

Fraser, G. E. and Stram, D. O. (2012). Regression calibration when foods (measured with error) are the variables of interest: markedly non-Gaussian data with many zeroes. *American journal of epidemiology* **175,** 325–31.

French, S. A., Mitchell, N. R., Wolfson, J., Harnack, L. J., Jeffery, R. W., Gerlach, A. F., Blundell, J. E., and Pentel, P. R. (2014). Portion size effects on weight gain in a free living setting. *Obesity (Silver Spring, Md.)* .

Freund, Y. and Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line

Learning and an Application to Boosting,. *Journal of Computer and System Sciences* **55,** 119–139.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive Logistic Regression: A Statistical View of Boosting. *The Annals of Statistics* **28,** 337–374.

Friedman, J. H. (2004). Gradient Directed Regularization. *Solutions* **2004,** 1–30.

Friedman, J. H. and Roosen, C. B. (1995). An introduction to multivariate adaptive regression splines. *Statistical Methods in Medical Research* **4,** 197–217.

Fuller, W. A. (1987). *Measurement Error Models.* Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA.

He, Z., Tu, W., Wang, S., Fu, H., and Yu, Z. (2015). Simultaneous variable selection for joint models of longitudinal and survival outcomes. *Biometrics* **71,** 178–87.

Hocking, R. (1976). The analysis and selection of variables in linear regression. *Biometrics* **32,** 1–49.

Hoerl, A. A. E. and Kennard, R. W. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* **12,** 55–67.

Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. (1999). Bayesian Model Averaging: A Tutorial. *Statistical Science* **14,** 382–401.

Huang, J., Sun, T., Ying, Z., Yu, Y., and Zhang, C. H. (2013). Oracle inequalities for the lasso in the cox model. *Annals of Statistics* **41,** 1142–1165.

Janes, H., Frahm, N., DeCamp, A., Rolland, M., Gabriel, E., Wolfson, J., Hertz, T., Kallas, E., Goepfert, P., Friedrich, D. P., Corey, L., Mullins, J. I., McElrath, M. J., and Gilbert, P. (2012). MRKAd5 HIV-1 Gag/Pol/Nef vaccine-induced T-cell responses inadequately predict distance of breakthrough HIV-1 sequences to the vaccine or viral load. *PloS one* **7,** e43396.

Johnson, B. A., Lin, D. Y., and Zeng, D. (2008). Penalized Estimating Functions and Variable Selection in Semiparametric Regression Models. *Journal of the American Statistical Association* **103,** 672–680.

Kipnis, V., Subar, A. F., Midthune, D., Freedman, L. S., Ballard-Barbash, R., Troiano, R. P., Bingham, S., Schoeller, D. A., Schatzkin, A., and Carroll, R. J. (2003). Structure of dietary measurement error: results of the OPEN biomarker study. *American journal of epidemiology* **158,** 14–21; discussion 22–6.

Koch, B.; Vock, V.; Wolfson, J. (2017). Covariate selection with group lasso and doubly robust estimation of causal effects. *Biometrics* .

Li, L., Shao, J., and Palta, M. (2005). A Longitudinal Measurement Error Model with a Semicontinuous Covariate. *Biometrics* **61,** 824–830.

Liang, K.-Y. and Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika* **73,** 13–22.

Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest**2,** 18–22.

Loh, P.-L. and Wainwright, M. J. (2012). High-dimensional regression with noisy and missing data: Provable guarantees with nonconvexity. *The Annals of Statistics* **40,** 1637–1664.

Ma, Y. and Li, R. (2010). Variable selection in measurement error models. *Bernoulli* **16,** 274–300.

MacKay, D. J. C. (1995). Developments in Probabilistic Modelling with Neural Networks Ensemble Learning. In *Neural Networks: Artificial Intelligence and Industrial Applications*, pages 191–198. Springer London, London.

Merz, C. J., Stolfo, S., Chan, P., and Wolpert, D. (1999). Using Correspondence Analysis to Combine Classifiers. *Machine Learning* **36,** 33–58.

Nakamura, T. (1990). Corrected score function for errors-in-variables models: Methodology and application to generalized linear models. *Biometrika* **77,** 127–137.

Ni, X., Zhang, D., and Zhang, H. H. (2010). Variable Selection for Semiparametric Mixed Models in Longitudinal Studies. *Biometrics* **66,** 79–88.

Pan, W. (2001). Akaike's Information Criterion in Generalized Estimating Equations. *Biometrics* **57,** 120–125.

Park, M. Y. and Hastie, T. (2007). L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **69,** 659–677.

Park, M. Y., Hastie, T., Young, M., and Hastie, P. T. (2006). L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **69,** 659–677.

Pike, F. and Weissfeld, L. (2013). Joint modeling of censored longitudinal and event time data. *Journal of Applied Statistics* **40,** 17–27.

R Core Team (2015). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Rosner, B., Spiegelman, D., and Willett, W. C. (1992). Correction of logistic regression relative risk estimates and confidence intervals for random within-person measurement error. *American journal of epidemiology* **136,** 1400–13.

Rosset, S. and Zhu, J. (2007). Piecewise linear regularized solution paths. *The Annals of Statistics* **35,** 1012–1030.

Rosset, S., Zhu, J., and Hastie, T. (2004). Boosting as a Regularized Path to a Maximum Margin Classifier. *Journal of Machine Learning Research* **5,** 941–973.

Ruczinski, I., Kooperberg, C., and Leblanc, M. (2003). Logic Regression. *Journal of Computational and Graphical Statistics* .

Schapire, R. E. (2003). The Boosting Approach to Machine Learning: An Overview. pages 149–171. Springer New York.

Schelldorfer, J., Meier, L., and Bühlmann, P. (2014). Glmmlasso: An algorithm for high-dimensional generalized linear mixed models using l_1-penalization. *Journal of Computational and Graphical Statistics* **23,** 460–477.

Sinisi, S. E. and van der Laan, M. J. (2004). Deletion/Substitution/Addition Algorithm in Learning with Applications in Genomics. *Statistical Applications in Genetics and Molecular Biology* **3,** 1–38.

Small, C. G. and Wang, J. (2003). *Numerical Methods for Nonlinear Estimating Equations.* Clarendon Press - Oxford.

Sørensen, Ø., Frigessi, A., and Thoresen, M. (2012). Measurement Error in Lasso: Impact and Correction. *arXiv.org* .

Spiegelman, D., McDermott, A., and Rosner, B. (1997). Regression calibration method for correcting measurement-error bias in nutritional epidemiology. *The American journal of clinical nutrition* **65,** 1179S–1186S.

Stefanski, L. A. and Carroll, R. J. (1985). Covariate Measurement Error in Logistic Regression. *The Annals of Statistics* **13,** 1335–1351.

Symonds, M. R. E. and Moussalli, A. (2011). A brief guide to model selection, multimodel inference and model averaging in behavioural ecology using Akaike's information criterion. *Behavioral Ecology and Sociobiology* .

Therneau, T. M., Atkinson, E. J., and Foundation, M. (1997). An Introduction to Recursive Partitioning Using the RPART Routines.

Tibshirani, R. (1994). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B* **58,** 267–288.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B* **58,** 267–288.

Todorovski, L.; Deroski, S. (2003). Combining Classifiers with Meta Decision Trees. *Machine Learning* **50,** 223–249.

Ueki, M. (2009). A note on automatic variable selection using smooth-threshold estimating equations. *Biometrika* **96,** 1005–1011.

Van De Geer, S. A. (2008). High-dimensional generalized linear models and the Lasso. *Annals of Statistics* **36,** 614–645.

Van Der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). Super Learner. *U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 222* .

Viallefont, V., Raftery, A. E., and Richardson, S. (2001). Variable selection and Bayesian model averaging in case-control studies. *Statistics in Medicine* **20,** 3215–3230.

Wasserman, L. (2004). *All of Statistics: A Concise Course in Statistical Inference.* Springer, New York.

Wolfson, J. (2011). EEBoost: A General Method for Prediction and Variable Selection Based on Estimating Equations. *Journal of the American Statistical Association* **106,** 296–305.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks* **5,** 241–259.

Yang, X., Belin, T. R., and Boscardin, W. J. (2005). Imputation and Variable Selection in Linear Regression Models with Missing Covariates. *Biometrics* **61,** 498–506.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *J. R. Statist. Soc. B* **68,** 49–67.

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

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *J. R. Statist. Soc. B* **67,** 301–320.