New Algorithms in Machine Learning with Applications in Personalized Medicine

by

Ying Daisy Zhuo

B.A., Middlebury College (2012)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author .....
Sloan School of Management
May 18, 2018

Certified by .....
Dimitris Bertsimas
Boeing Leaders for Global Operations Professor
Co-Director, Operations Research Center
Thesis Supervisor

Accepted by .....
Patrick Jaillet
Dugald C. Jackson Professor
Department of Electrical Engineering and Computer Science
Co-Director, Operations Research Center

New Algorithms in Machine Learning with Applications in Personalized Medicine

by

Ying Daisy Zhuo

Submitted to the Sloan School of Management
on May 18, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Recent advances in machine learning and optimization hold much promise for influencing real-world decision making, especially in areas such as health care where abundant data are increasingly being collected. However, imperfections in the data pose a major challenge to realizing their full potential: missing values, noisy observations, and unobserved counterfactuals all impact the performance of data-driven methods.

In this thesis, with a fresh perspective from optimization, I revisit some of the well-known problems in statistics and machine learning, and develop new methods for prescriptive analytics. I show examples of how common machine learning tasks, such as missing data imputation in Chapter 2 and classification in Chapter 3, can benefit from the added edge of rigorous optimization formulations and solution techniques. In particular, the proposed `opt.impute` algorithm improves imputation quality by 13.7% over state-of-the-art methods, as averaged over 95 real data sets, which leads to further performance gains in downstream tasks. The power of prescriptive analytics is shown in Chapter 4 by our approach to personalized diabetes management, which identifies response patterns using machine learning and individualizes treatments via optimization.

These newly developed machine learning algorithms not only demonstrate improved performance in large-scale experiments, but are also applied to solve the problems in health care that motivated them. Our simulated trial for diabetic patients in Chapter 4 demonstrates a clinically relevant reduction in average hemoglobin A1c levels compared to current practice. Finally, when predicting mortality for cancer patients in Chapter 5, applying `opt.impute` on missing data along with the cutting-edge algorithm *Optimal Classification Tree* on a rich data set prepared from electronic medical records, we are able to accurately risk stratify patients, providing physicians with interpretable insights and valuable risk estimates at time of treatment decisions and end-of-life planning.

Thesis Supervisor: Dimitris Bertsimas
Title: Boeing Leaders for Global Operations Professor
Co-Director, Operations Research Center

Acknowledgments

First and foremost, I am incredibly fortunate to have Professor Dimitris Bertsimas as my academic and thesis advisor. His vision on making an impact in the world through research has shaped my aspirations over the years, and will continue to influence me in the future. I cannot thank him enough for caring so deeply about my personal and professional development as an advisor, a mentor, and a friend.

I want to thank Professor Nikos Trichakis and Professor Jónas Jónasson for kindly agreeing to be on my thesis committee and providing helpful suggestions in the thesis writing process, and would like to thank Professor Robert Freund and Professor Rahul Mazumder for being on my general exam committee and providing me with valuable advice and feedback on research early on.

This work would not have been possible without the support from my collaborators at the ORC. My close friend and colleague Colin Pawlowski and I spent countless hours working together on many projects, three of which made into this thesis. His contribution to this cannot be overemphasized. Thanks to Jack Dunn for always motivating me to work hard and work efficiently, teaching me disciplined coding practice, and continuing to encourage and support me as a friend. I am grateful to Alex Weinstein and Nathan Kallus in our collaboration on the personalized diabetes management paper, whose creativity and perseverance have made the project successful. Agni and Holly, your positivity as we continue with new research ideas will prove to accelerate the success of the projects, and I am very much looking forward to seeing it happen.

I want to say special thanks to my other friends at the ORC, especially to Yeesian, whose kindness and wisdoms continue to influence me and make me a better person. I will greatly miss our coffee times together. Many thanks to Emma, Nishanth, Julia, Kevin, Jing, Yuchen, Rim and many others who made my time at the ORC special.

I would like to express my gratitude to Dr. Aymen Elfiky and Dr. Eddy Chen in their long-term support and offering of their medical expertise in the preparation of the cancer mortality project. In support of "Personalized Diabetes Management",

thanks are due to Dr. Michael Kane of MIT Medical for sharing clinical expertise in the progression and treatment of diabetes, and to Dr. William Adams of Boston University for assisting with the interpretation of the medical records.

Finally, I want to thank my parents for allowing their only child to go on an adventure in a country far way, trusting that I have been working toward a doctorate degree from an accredited university. I am also grateful to my parents-in-law, Cindy and Ollie Curme, who have never stopped giving me love and support. Lastly, my deepest and greatest thanks are due to my husband Chester Curme, without whom I would not have been so brave to embark on this exciting journey.

Contents

Chapter 1

Introduction

1.1 Background

Despite advances in many areas of medicine, patient treatment has still largely been one-size-fits-all. This is in part due to the lack of large-scale evidence based studies that compare treatment effects on the individual level. In fact, historically almost all clinical trials were designed to test the average treatment effects, rather than individual effects [83]. However, mounting clinical evidence shows that patients exhibit differential responses to drugs. In oncology, for example, recent research has suggested that patients with specific genetic mutations have much higher response rates to certain targeted therapies [41, 62], which leads to the development of a generation of new therapies, and a national push toward more precision-based cancer medicine.

In response to the need for better individual-level evidence and regulatory requirements, large-scale patient data are increasingly being collected and digitized. It is often believed that the vast amount of data, if used appropriately, can streamline care delivery for providers, improve health outcomes for patients, and reduce costs to the system for payers [79]. Combined with the right analytic tools, promising results have been observed in academic research in fields such as machine learning and operations research [16]. Nevertheless, clinical adoption has been low, especially in areas of personalized treatment recommendations.

Health data are becoming increasingly complex as they rapidly evolve. The

widespread use of electronic health records (EHR), high-dimensional genomic testing results, raw imagining, physician notes, and even personal health tracker data, brings new challenges in utilizing the data effectively. Because many of these data sets are not carefully curated by humans, as in the cases of traditional clinical trial data, new issues such as large-scale missingness, outliers, and errors in the entries appear throughout health data. In addition, the nature of retrospective observational data inevitably invites issues in confounding variables and the lack of counterfactual outcomes when one tries to compare effectiveness and draw inference. As a result, to obtain clinically relevant insights in the presence of these challenges, new machine learning algorithms must be designed to be robust in such scenarios.

With the overarching goal of advancing data-driven personalized medicine, my focus in the areas of machine learning algorithms has largely been influenced by the challenges that medical data analytics have been facing. In particular, as missing data is prevalent across disease registry, EHR, and genomic data sources, in Chapter 2 I develop a novel missing data imputation method that produces highly accurate imputations, which lead to superior performance in downstream machine learning tasks. In Chapter 3, the data uncertainties in both the training features and labels in classification tasks (for example, mortality prediction) are addressed via robust formulations of common classifiers with demonstrated improvement in prediction accuracy.

The next two chapters work directly with medical data. Chapter 4 explores the personalization of type 2 diabetes based on a rich EHR data set, using a novel prescriptive k-nearest neighbors approach that estimates counterfactual outcomes and prescribes optimal treatments. This method demonstrated improved clinical outcomes under our estimation. Finally in Chapter 5, I present a highly accurate mortality prediction model among cancer patients based on EHR data from a large national cancer center, using the imputation method developed in Chapter 2 and a novel machine learning method *Optimal Classification Trees*.

1.2 Main Contributions

The major contributions in this thesis can be summarized as follows, listed by chapter.

Chapter 2. From Predictive Methods to Missing Data Imputation

- **New formulations under a general framework**

  We pose the missing data problem under a general optimization framework using a predictive model-based cost function, with examples derived from K-nearest neighbors, support vector machines, and optimal decision tree models. This optimization perspective provides fresh insight into the classical missing data problem and leads to new algorithms for more accurate data imputation.

- **Fast and scalable first-order methods**

  For each imputation model, we derive first-order methods to find high-quality solutions to the missing data problem following a general imputation algorithm `opt.impute`. These methods easily scale to large data sets with convergence within a few iterations. In addition, the first-order methods are robust and reliable for arbitrary missing patterns and mixed data types.

- **Superior imputation accuracy**

  We evaluate the methods in computational experiments using 95 real-world data sets taken from the UCI Machine Learning Repository. Benchmarked against multiple state-of-the-art imputation methods, `opt.impute` produces the best overall imputation in more than 74.9% of all data sets averaged over all missing data scenarios considered.

- **Improved performance in downstream tasks**

  We demonstrate that the data imputations generated by `opt.impute` give rise to substantial improvement in the performance on downstream classification and regression tasks.

Chapter 3. Robust Classification

- **Principled framework for robust classification**

  We present a framework to robustify existing classifiers in a unified and principled way, with an aim to build classifiers that model data uncertainties in features, labels, as well as both features and labels simultaneously. This leads to tractable problems with relatively small overhead compared to the original methods. In particular, we use this framework to derive counterparts to SVM, logistic regression, and CART that are robust to variations in features and labels in the data.

- **Better hyperplane recovery in synthetic experiments**

  We demonstrate the advantage of robust formulations over regularized and nominal methods through synthetic data experiments with two classes divided by a separating hyperplane. Compared to nominal and regularized methods, the robust SVM and logistic regression methods recover the separating hyperplane classifiers closer to the truth, leading to gains in out-of-sample accuracy especially in the worst case analysis.

- **Improved out-of-sample performance in real-world data sets**

  We demonstrate that robust classification improves out-of-sample accuracy in large-scale computational experiments across a sample of 75 data sets from the UCI Machine Learning Repository. Furthermore, we identify characteristics of classification problems for which robust methods lead to significant accuracy gains compared to non-robust methods. Specifically, in problems with high dimensional data and difficult separability, the value of robustness is even more prominent.

- **Empirical rule for when to use robust classifiers**

  We provide a simple, empirically-derived decision rule for machine learning practitioners that predicts with high accuracy when robust methods can offer significant improvement over the nominal methods, with an average improvement in out-of-sample accuracy of 5.3% for SVM, 4.0% for logistic regression, and 1.3% for CART. Compared to regularized SVM or logistic regression, the average

out-of-sample accuracy improvement of our principled approach to robustness is 2.1% over regularized SVM and 1.2% over regularized logistic regression when this rule is satisfied.

Chapter 4. Personalized Diabetes Management Using Electronic Medical Records

- **First of its kind: Personalized treatment recommender for diabetes**
  Our study is the first of its kind to use machine learning to develop an algorithm for personalized treatment recommendations using electronic medical records. The algorithm can be integrated into existing EMR systems to dynamically suggest personalized treatment paths for each patient based on historical records.

- **Improved HbA1c level relative to current practice**
  Based on simulations with data from Boston Medical Center, we estimate that the use of our algorithm could improve outcomes for patients with type 2 diabetes by reducing post-treatment glycated hemoglobin levels significantly relative to current practice.

- **Interactive dashboard with intuitive rationale behind recommendations**
  We prototype an intuitive, interactive dashboard that summarizes the evidence for each recommendation, including the expected distribution of outcomes under alternative treatments. We believe this integrated, interactive approach has the potential to reshape medical practice across the disease spectrum.

Chapter 5. An Actionable Tool for Mortality Predictions in Cancer Patients

- **Personalized and specific**
  The tool takes as inputs the EHR of a particular patient, the cancer type and an

envisioned cancer treatment and outputs the mortality risk adjusted for these patient characteristics.

- **Interpretable and clinically meaningful**

  A physician can easily understand the reasoning behind the algorithm, illustrated as an interpretable decision tree. The model also identifies key predictors of mortality such as change in weight.

- **Evidence-based and data-driven**

  The tool was informed by EHRs of more than 23,000 patients at a large national cancer hospital. We included 401 predictors including demographics, medical and treatment history, laboratory tests, and genomic results.

- **Actionable**

  The clinician can compare different envisioned treatments for a particular patient with respect to the range of mortality risk and make decisions informed by these estimates.

- **Validated and accurate**

  We compare the accuracy and the area under the curve (AUC) in unseen patient data from 2012-2014, with very encouraging results compared to competing approaches.

- **Based on novel development in modern machine learning**

  The methodology of this study is based on two novel algorithms: a) the predictive tree developed using optimization ideas, [7] and b) the statistical method for missing data imputation introduced in Chapter 2 [18].

Chapter 2

From Predictive Methods to Missing Data Imputation: An Optimization Approach

This work, co-authored with Dimitris Bertsimas and Colin Pawlowski, is accepted to the Journal of Machine Learning Research [18].

Missing data is a common problem in real-world settings and for this reason has attracted significant attention in the statistical literature. We propose a flexible framework based on formal optimization to impute missing data with mixed continuous and categorical variables. This framework can readily incorporate various predictive models including K-nearest neighbors, support vector machines, and decision tree based methods, and can be adapted for multiple imputation. We derive fast first-order methods that obtain high quality solutions in seconds following a general imputation algorithm `opt.impute` presented here. We demonstrate that our proposed method improves out-of-sample accuracy in large-scale computational experiments across a sample of 95 data sets taken from the UCI Machine Learning Repository. In all scenarios (missing completely at random or not, under various missing percentages), `opt.impute` produces the best overall imputation in most data sets benchmarked against five other methods: mean impute, K-nearest neighbors, it-

erative knn, Bayesian PCA, and predictive-mean matching, with an average reduction in mean absolute error of 13.7% from the second best method. Moreover, `opt.impute` leads to improved out-of-sample performance of learning algorithms trained using the imputed data, demonstrated by computational experiments on 10 downstream tasks. For models trained using `opt.impute` single imputations with 50% data missing, the average out-of-sample R^2 is 0.356 in the regression tasks and the average out-of-sample accuracy is 86.6% in the classification tasks, compared to 0.312 and 85.2% for the best benchmark single imputation methods. In the multiple imputation setting, regression models trained using `opt.impute` outperform models trained using multivariate imputation by chained equations (`mice`) in 84% of missing data scenarios considered.

2.1 Introduction

The missing data problem is arguably the most common issue encountered by machine learning practitioners when analyzing real-world data. In many applications ranging from gene expression in computational biology to survey responses in social sciences, missing data is present to various degrees. As many statistical models and machine learning algorithms rely on complete data sets, it is key to handle the missing data appropriately.

In some cases, simple approaches may suffice to handle missing data. For example, complete-case analysis uses only the data that is fully known and omits all observations with missing values to conduct statistical analysis. This works well if only a few observations contain missing values, and when the data is missing completely at random, complete-case analysis does not lead to biased results [69]. Alternately, some machine learning algorithms naturally account for missing data, and there is no need for preprocessing. For instance, CART and K-means have been adapted for problems with missing data [29, 95].

In many other situations, missing values need to be imputed prior to running statistical analyses on the complete data set. The benefit of the latter approach is

Table 2.1: List of Imputation Methods

Method Name	**Category**	**Software**	**Reference**
Mean impute (`mean`)	Mean		[69]
Expectation-Maximization (`EM`)	EM		[40]
EM with Mixture of Gaussians and Multinomials	EM		[50]
EM with Bootstrapping	EM	`Amelia II`	[56]
K-Nearest Neighbors (`knn`)	K-NN	`impute`	[92]
Sequential K-Nearest Neighbors	K-NN		[65]
Iterative K-Nearest Neighbors	K-NN		[34, 27]
Support Vector Regression	SVR		[96]
Predictive-Mean Matching (`pmm`)	LS	`MICE`	[32]
Least Squares	LS		[26]
Sequential Regression Multivariate Imputation	LS		[78]
Local-Least Squares	LS		[64]
Sequential Local-Least Squares	LS		[101]
Iterative Local-Least Squares	LS		[33]
Sequential Regression Trees	Tree	`MICE`	[31]
Sequential Random Forest	Tree	`missForest`	[85]
Singular Value Decomposition	SVD		[92]
Bayesian Principal Component Analysis	SVD	`pcaMethods`	[76, 73]
Factor Analysis Model for Mixed Data	FA		[63]

that once a set (or multiple sets) of complete data has been generated, practitioners can easily apply their own learning algorithms to the imputed data set. We focus on methods for missing data imputation in this work.

Concretely, assume that we are given data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with missing entries $x_{id}, (i, d) \in \mathcal{M}$. The objective is to impute the values of the missing data that resemble the underlying complete data as closely as possible. This way, when one conducts statistical inference or pattern recognition using machine learning methods on the imputed data, the results should be similar to those obtained if full data were given. We outline some of the state-of-the-art methods for imputation in Table 2.1 and describe them briefly below. Part of the list is adapted from a review paper by Liew et al. [67].

The simplest method is mean impute, in which each missing value x_{id} is imputed as the mean of all observed values in dimension d. Mean impute underestimates the variance, ignores the correlation between the features, and thus often leads to poor imputation [69].

Joint modeling asserts some joint distribution on the entire data set. It assumes a parametric density function (e.g., multivariate normal) on the data given model

parameters. In practice, model parameters are typically estimated using an Expectation-Maximization (EM) approach. It finds a solution (often non-optimal) of missing values and model parameters to maximize the likelihood function. Many software tools such as the R package `Amelia 2` implement the EM method with bootstrapping, assuming that the data is drawn from a multivariate normal distribution [56]. Joint modeling provides useful theoretical properties but lacks the flexibility for processing data types seen in many real applications [93]. For example, when the data includes continuous and categorical variable types, standard multivariate density functions often fail at modeling the complexity of mixed data types. However, under the assumption that the categorical variables are independent, we can use mixture models of Gaussians and Multinomials for imputation [50].

In contrast to joint modeling, fully conditional specification is a more flexible alternative where one specifies the conditional model for each variable; it is especially useful in mixed data types [93]. To generalize to multivariate settings, a chained equation process – initializing using random sampling and conducting univariate imputations sequentially until convergence – is typically used [32]. Each iteration is a Gibbs sampler that draws from the conditional distribution on the imputed values.

A simple example of conditional specification is based on regression. Least-Squares (LS) imputation constructs single univariate regressions, regressing features with missing values on all of the other dimensions in the data. Each missing value x_{id} is then imputed as the weighted average of these regression predictions [26, 78]. Alternatively, in the Predictive-Mean Matching method (`pmm`), imputations are random samples drawn from a set of observed values close to regression predictions [32]. Imputation methods that use Support Vector Regression in place of LS for the regression step have also been explored [96].

When there is non-linear relationship between the variables, linear regression based imputation may perform poorly. [31] propose using Classification and Regression Trees (CART) as the conditional model for imputation. Extensions to random forests have also shown promising results [85]. These decision tree based imputation methods are non-parametric approaches that do not rely upon distributional

assumptions on the data.

One of the most commonly used non-parametric approaches is K-Nearest Neighbors (K-NN) based imputation. This method imputes each missing entry x_{id} as the mean of the dth dimension of the K-nearest neighbors that have observed values in dimension d [92]. Some extensions of K-NN include sequential K-NN, which starts by imputing missing values from observations with the fewest missing dimensions and continues imputing the next unknown entries reusing the previously imputed values [65]. Iterative K-NN uses an iterative process to refine the estimates and choose the nearest neighbors based on the estimates from the previous iteration [34, 27]. The Local-Least Squares method combines ideas from K-NN and LS, imputing each missing value x_{id} using regression models trained on the K-nearest neighbors of the point $\mathbf{x}_i$ [64]. Sequential and iterative variations of Local-Least Squares resemble their K-NN imputation counterparts [101, 33].

Low dimensional representation-based imputation assumes that the data represents a noisy observation of a linear combination of a small set of principal components or factor variables. In the basic method, singular value decomposition (SVD) is used on the entire data set to determine the principal eigenvectors. The missing values are imputed as a linear combination of these eigenvectors. This process is iteratively repeated until convergence [92, 72]. Bayesian Principal Component Analysis is similar to SVD imputation but extends the method to incorporate information from a prior distribution on the model parameters [76, 73]. Some recent development of a variant of the EM algorithm for factor analysis also provides a missing data imputation method for mixed data [63].

Thus far, we have only discussed methods for single imputation which generate one set of completed data that will be used for further statistical analyses. Multiple imputation, on the other hand, imputes multiple times (each set is possibly different), runs the statistical analyses on each, and pools the results [69]. Such method is able to capture the variability in the missing data and therefore generate potentially more accurate estimates to the larger statistical problem. However, multiple imputation methods are slower and require pooling results, which may not be appropriate for

certain applications.

Within the multiple imputation framework, the procedure for generating multiple estimates of missing values varies. Multivariate imputation by chained equations (`mice`), a popular multiple imputation method, generates estimates using: predictive mean matching, Bayesian linear regression, logistic regression, and others [32]. In all cases, the method initializes using random sampling and conducts univariate imputations sequentially until convergence. Each iteration is a Gibbs sampler that draws from the conditional distribution on the imputed values.

Because of its importance, missing data imputation remains an active research area. Although there are numerous methods, many of them have serious shortcomings. Joint modeling methods are not as effective when data sets violate normality assumptions, and a naïve implementation often crashes during the computation of a singular covariance matrix [56]. Some conditional specification methods such as `pmm` are practically reliable, but lack theoretical foundation and have no explicit formulation as an optimization problem. This stands in stark contrast to other areas of machine learning, where statistical models and optimization problems are deeply intertwined.

Evidence from recent literature suggests that recent advances in optimization have driven significant progress in machine learning. Integer and convex optimization have been applied successfully to median and sparse regression problems [21, 15]. Recent work on Optimal Decision Trees for classification leverages integer and robust optimization [7, 9]. In this work, we reconsider the missing data problem from this perspective, in order to develop optimization-based methods for imputation with improved out-of-sample performance.

Contributions We summarize our contributions below:

1. We pose the missing data problem under a general optimization framework. The framework produces an optimization problem with a predictive model-based cost function that explicitly handles both continuous and categorical variables and can be used to generate multiple imputations. We present three cost func-

tions derived from K-nearest neighbors, support vector machines, and optimal decision tree models. This optimization perspective provides fresh insight into the classical missing data problem and leads to new algorithms for more accurate data imputation.

2. For each imputation model, we derive first-order methods to find high-quality solutions to the missing data problem following a general imputation algorithm `opt.impute` presented here. These methods easily scale to data sets with n in the 100,000s and p in the 1,000s on a standard desktop computer and converge within a few iterations. In addition, the first-order methods are robust and reliable for arbitrary missing patterns and mixed data types.

3. We evaluate the methods in computational experiments using 95 real-world data sets taken from the UCI Machine Learning Repository. Benchmarked against existing imputation methods including mean impute, K-nearest neighbors, iterative knn, Bayesian PCA, and predictive-mean matching, `opt.impute` produces the best overall imputation in more than 75.8% of all data sets, and results in an average reduction in mean absolute error of 8.3% against the best cross-validated benchmark method.

4. We demonstrate that the improved data imputations generated by `opt.impute` give rise to improved performance on 10 downstream classification and regression tasks. With 50% of missing data, classification models trained on data imputed via `opt.impute` have an average testing accuracy of 86.1% compared to 84.4% for the best cross-validated benchmark method. In addition, regression models trained on data imputed via `opt.impute` have an average out-of-sample R^2 value of 0.339 compared to 0.315 for the best cross-validated benchmark method. Finally, downstream models trained on multiple imputations produced by `opt.impute` significantly outperform multiple imputations produced by `mice` in 3/5 missing data scenarios for classification and 5/5 scenarios for regression.

The structure of the chapter is as follows. In Section 2.2, we formulate the missing data imputation problem as an optimization problem, present a general first-order

method `opt.impute` that can be used to find high-quality solutions, and derive the algorithms for each model: K-NN, SVM, and trees. We also discuss a cross-validation procedure and extensions of `opt.impute` to multiple imputation. In Section 2.3, we compare the imputation quality and performance on downstream tasks of `opt.impute` to benchmark imputation methods on a wide range of real data sets. In Section 2.4, we discuss the benefits from adopting such framework and suggest areas for future work. We conclude in Section 2.5.

2.2 Methods for Optimal Imputation

In this section, we pose the missing data problem as an optimization problem in which we optimize the missing values in all data points and dimensions simultaneously. We introduce a general imputation framework on mixed data (continuous and categorical) based upon first-order methods applied to this problem. Within this framework, we use K-nearest neighbors, SVM, and decision tree based imputation as examples to define three specific optimization problems. For each problem, we present two first-order methods used to find high-quality solutions: block coordinate descent (BCD) and coordinate descent (CD).

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ be the dataset given with p variables. Without loss of generality, we assume each data vector $\mathbf{x}_i$ contains continuous variables indexed by $d \in \{1, 2, \ldots, p_0\}$ and categorical variables indexed by $d \in \{p_0 + 1, \ldots, p_0 + p_1\}$ with $p_0 + p_1 = p$. As a pre-processing step, we transform all continuous variables to have unit standard deviation. We leave all categorical variables unchanged, and assume the dth categorical variable $d \in \{p_0 + 1, \ldots, p_0 + p_1\}$ takes values among k_d classes. Note that if all data is continuous $p_0 = 0$, while if all data is categorical

$p_1 = 0$. The missing and known values are specified by the following sets:

$$\begin{aligned}
\mathcal{M}_0 &= \{(i,d) : \text{entry } x_{id} \text{ is missing, } 1 \le d \le p_0\},\\
\mathcal{N}_0 &= \{(i,d) : \text{entry } x_{id} \text{ is known, } 1 \le d \le p_0\},\\
\mathcal{M}_1 &= \{(i,d) : \text{entry } x_{id} \text{ is missing, } p_0+1 \le d \le p_0+p_1\},\\
\mathcal{N}_1 &= \{(i,d) : \text{entry } x_{id} \text{ is known, } p_0+1 \le d \le p_0+p_1\}.
\end{aligned}$$

We also refer to the full missing pattern as $\mathcal{M} := \mathcal{M}_0 \cup \mathcal{M}_1$. Let $\mathbf{W} \in \mathbb{R}^{n \times p_0}$ be the matrix of imputed continuous values, where w_{id} is the imputed value for entry x_{id}, $d \in \{1, \ldots, p_0\}$. Similarly, let $\mathbf{V} \in \{1, \ldots, k_1\} \times \ldots \times \{1, \ldots, k_{p_1}\}$ be the matrix of imputed categorical values, where v_{id} is the imputed value for entry x_{id}, $d \in \{p_0+1, \ldots, p_0+p_1\}$. We refer to the full imputation for observation $\mathbf{x}_i$ as $(\mathbf{w}_i, \mathbf{v}_i)$ in the following sections.

2.2.1 General Problem Formulation

As the task is to impute the missing values, for each model the key decision variables are the imputed values $\{w_{id} : (i,d) \in \mathcal{M}_0\}$ and $\{v_{id} : (i,d) \in \mathcal{M}_1\}$. We also introduce auxiliary decision variables as well; denote these as $\mathbf{U}$. For instance, in a K-NN based approach, indicator variables $z_{ij}, 1 \le i, j \le n$ are introduced to identify the neighbor assignment for each pair of points $\mathbf{x}_i$, $\mathbf{x}_j$. For a given set of imputed values and a given model, there is a cost function $c(\cdot)$ associated with it. Our goal is to solve the following optimization problem:

$$\begin{aligned}
\min \quad & c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X}) \\
\text{s.t.} \quad & w_{id} = x_{id} \quad (i,d) \in \mathcal{N}_0, \\
& v_{id} = x_{id} \quad (i,d) \in \mathcal{N}_1, \\
& (\mathbf{U}, \mathbf{W}, \mathbf{V}) \in \mathcal{U},
\end{aligned} \tag{2.1}$$

where $\mathcal{U}$ is the set of all feasible combinations $(\mathbf{U}, \mathbf{W}, \mathbf{V})$ of auxiliary vectors and imputations. For example, in a K-NN based approach, this includes the constraints

that each point has exactly K neighbors and the assignment variables are binary. We list the auxiliary variables and cost functions corresponding to each of the imputation models K-NN, SVM, and trees in Table 2.2. Note that the cost function can be different for continuous and categorical variables. We can introduce a parameter that controls the relative contribution to the cost between the continuous and categorical variables, or scale continuous variables appropriately. For the remainder of the chapter the latter is assumed for simplicity of notation.

Table 2.2: Variables and cost functions for each imputation model. Variables for K-NN, SVM, and trees are defined in Sections 2.2.3, 2.2.4, and 2.2.5 respectively.

Model	**U**	$\mathbf{c}(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X})$
K-NN	$\mathbf{Z}$	$\sum_{i\in\mathcal{I}}\sum_{j=1}^{n} z_{ij}\Big[\sum_{d=1}^{p_0}(w_{id}-w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1}\mathbb{1}_{\{v_{id}\neq v_{jd}\}}\Big]$
SVM	$[\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\xi}]$	$\frac{1}{2}(\|\boldsymbol{\beta}\|_{\mathcal{H}}^2 + \|\boldsymbol{\theta}\|_{\mathcal{H}}^2) + C\sum_{i=1}^{n}\left(\sum_{d=1}^{p_0}(\gamma_{id}+\gamma_{id}^*) + \sum_{d=p_0+1}^{p_0+p_1}\xi_{id}\right)$
Trees	$\mathbf{T}$	$\sum_{i=1}^{n}\sum_{j=1}^{n}\Big[\sum_{d=1}^{p_0} t_{ij}^d(w_{id}-w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} t_{ij}^d\mathbb{1}_{\{v_{id}\neq v_{jd}\}}\Big]$

This problem is non-convex for K-NN, SVM, and tree models. To obtain a certifiable optimal solution, one can reformulate the problem with integer variables and solve it using a mixed integer solver. We ran computational experiments and found that solving such mixed integer problems requires a long time to reach a certifiably optimal solution. As a result, we present a general imputation algorithm `opt.impute` which approximates the solution to Problem (2.1) very fast using first-order methods.

2.2.2 First-Order Method for the General Problem

To obtain high-quality solutions to problem (2.1), we can use first-order methods with random warm starts. In particular, we will focus on block coordinate descent (BCD) and coordinate descent (CD) [4]. Algorithm 1, which we refer to as `opt.impute`, implements BCD or CD for Problem (2.1). The variables $\mathbf{U}, \mathbf{W}, \mathbf{V}$, and $\mathbf{X}$ as well as the cost function $c(\cdot)$ are summarized in Table 2.2 for K-NN, SVM, and trees. The detailed solution methods for problems (2.2), (2.3), (2.4), and (2.5) for K-NN, SVM,

and tree imputation models are described in Sections 2.2.3-2.2.5, respectively.

By construction, the objective function value strictly decreases by at least δ_0 until termination. It follows that the number of steps needed for the algorithm to terminate is $\lceil \frac{1}{\delta_0} c(\mathbf{U}^0, \mathbf{W}^0, \mathbf{V}^0; \mathbf{X}) \rceil$, where $\mathbf{W}^0, \mathbf{V}^0$ are the initialization values, $\mathbf{X}$ is data, and $\mathbf{U}^0$ is the argmin in Equation (2.2). However, the algorithm is not guaranteed to find a global minimum for Problem (2.1) [98].

In the next sections, we discuss three example models and the optimization problem formulations. For each model and each first-order method, we derive the specific updates for $\mathbf{U}, \mathbf{W}, \mathbf{V}$ that we use in our optimization-based imputation procedure. After, we describe a cross-validation procedure to select the specific model and parameters for the imputation.

2.2.3 K-NN Based Imputation

We first define a distance metric between rows $(\mathbf{w}_i, \mathbf{v}_i)$ and $(\mathbf{w}_j, \mathbf{v}_j)$ as

$$d_{ij} := \sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} \mathbb{1}_{\{v_{id} \neq v_{jd}\}}. \tag{2.6}$$

Next, we introduce the binary variables:

$$z_{ij} = \begin{cases} 1, & \text{if } (\mathbf{w}_j, \mathbf{v}_j) \text{ is among the } K\text{-nearest neighbors of } (\mathbf{w}_i, \mathbf{v}_i) \\ & \text{with respect to distance metric (2.6),} \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

Algorithm 1 `opt.impute`

Input: $\mathbf{X} \in \mathbb{R}^{n\times p_0} \times \{1,\ldots,k_1\} \times \ldots \times \{1,\ldots,k_{p_1}\}$,
a data matrix with some missing entries $\mathcal{M} = \{(i,d) : x_{id} \text{ is missing}\}$,
$\delta_0 > 0$, and warm start $\mathbf{W}^0 \in \mathbb{R}^{n\times p_0}$, $\mathbf{V}^0 \in \{1,\ldots,k_1\} \times \ldots \times \{1,\ldots,k_{p_1}\}$.

Output: $\mathbf{X}^{imp}$ a full matrix with imputed values.

Procedure:

Initialize $\delta \leftarrow \infty$, $\mathbf{W}^{old} \leftarrow \mathbf{W}^0$, $\mathbf{V}^{old} \leftarrow \mathbf{V}^0$.

while $\delta > \delta_0$ **do**

① Update $\mathbf{U}^*$ using model dependent information:

$$\begin{aligned} \mathbf{U}^* \leftarrow \arg\min_{\mathbf{U}} \ & c(\mathbf{U}, \mathbf{W}^{old}, \mathbf{V}^{old}; \mathbf{X}) \\ \text{s.t.} \ & (\mathbf{U}, \mathbf{W}^{old}, \mathbf{V}^{old}) \in \mathcal{U}. \end{aligned} \tag{2.2}$$

② Update the imputation $\mathbf{W}^*$, $\mathbf{V}^*$, following either:

(2a) block coordinate descent (BCD):

$$\begin{aligned} \mathbf{W}^*, \mathbf{V}^* \leftarrow \arg\min_{\mathbf{W},\mathbf{V}} \ & c(\mathbf{U}^*, \mathbf{W}, \mathbf{V}; \mathbf{X}) \\ \text{s.t.} \quad & w_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_1, \\ & (\mathbf{U}^*, \mathbf{W}, \mathbf{V}) \in \mathcal{U}, \end{aligned} \tag{2.3}$$

or

(2b) coordinate descent (CD):

$$\begin{aligned} w^*_{jr} \leftarrow \arg\min_{w_{jr}} \ & c(\mathbf{U}^*, \mathbf{W}, \mathbf{V}; \mathbf{X}) \\ \text{s.t.} \quad & w_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_1, \\ & w_{id} = w^*_{id} \qquad (i,d) \in \mathcal{M}_0 \backslash (j,r), \\ & v_{id} = v^*_{id} \qquad (i,d) \in \mathcal{M}_1, \\ & (\mathbf{U}^*, \mathbf{W}, \mathbf{V}) \in \mathcal{U}, \end{aligned} \tag{2.4}$$

$$\begin{aligned} v^*_{jr} \leftarrow \arg\min_{v_{jr}} \ & c(\mathbf{U}^*, \mathbf{W}, \mathbf{V}; \mathbf{X}) \\ \text{s.t.} \quad & w_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_1, \\ & w_{id} = w^*_{id} \qquad (i,d) \in \mathcal{M}_0, \\ & v_{id} = v^*_{id} \qquad (i,d) \in \mathcal{M}_1 \backslash (j,r), \\ & (\mathbf{U}^*, \mathbf{W}, \mathbf{V}) \in \mathcal{U}. \end{aligned} \tag{2.5}$$

③ $\delta \leftarrow c(\mathbf{U}^*, \mathbf{W}^*, \mathbf{V}^*; \mathbf{X}) - c(\mathbf{U}^{old}, \mathbf{W}^{old}, \mathbf{V}^{old}; \mathbf{X})$.

④ $(\mathbf{U}^{old}, \mathbf{W}^{old}, \mathbf{V}^{old}) \leftarrow (\mathbf{U}^*, \mathbf{W}^*, \mathbf{V}^*)$.

end while

$\mathbf{X}^{imp} \leftarrow [\mathbf{W}^*; \mathbf{V}^*]$

We further define the set of indices $\mathcal{I} := \{i : \mathbf{x}_i \text{ has at least one missing coordinate}\}$. The optimization problem for the K-NN based imputation model is:

$$\begin{aligned}
\min \quad & c(\mathbf{Z}, \mathbf{W}, \mathbf{V}; \mathbf{X}) := \sum_{i \in \mathcal{I}} \sum_{j=1}^{n} z_{ij} \Big[\sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} \mathbb{1}_{\{v_{id} \neq v_{jd}\}} \Big] \\
\text{s.t.} \quad & w_{id} = x_{id} && (i,d) \in \mathcal{N}_0, \\
& v_{id} = x_{id} && (i,d) \in \mathcal{N}_1, \\
& z_{ii} = 0, && i \in \mathcal{I}, \\
& \sum_{j=1}^{n} z_{ij} = K, && i \in \mathcal{I}, \\
& \mathbf{Z} \in \{0,1\}^{|\mathcal{I}| \times n}.
\end{aligned} \tag{2.8}$$

By optimality, it follows that $z_{ij} = 1$ if and only if $(\mathbf{w}_j, \mathbf{v}_j)$ is among the K-nearest neighbors of $(\mathbf{w}_i, \mathbf{v}_i)$. Therefore, solving problem (2.8) produces the missing value imputation which minimizes the sum of distances from each point $(\mathbf{w}_i, \mathbf{v}_i), i \in \mathcal{I}$ to its K-nearest neighbors. Note that the relation $\mathbb{1}_{\{v_{id} \neq v_{jd}\}}$ can be modeled with binary variables. Problem (2.8) is a nonconvex optimization problem with both continuous and binary variables. Correspondingly, it is difficult to solve to provable optimality, even if the data set contains continuous variables only.

Next, we describe the updates in Algorithm 1 for K-NN based imputation. We refer to this specific imputation method as `opt.knn`.

In step ①, to update the auxiliary variables $\mathbf{Z}$, first fix all imputed values $\mathbf{W}$, $\mathbf{V}$. Problem (2.2) decomposes by $i \in \mathcal{I}$ into the assignment problems:

$$\begin{aligned}
\min_{\mathbf{z}_i} \quad & \sum_{j=1}^{n} z_{ij} d_{ij} \\
\text{s.t.} \quad & z_{ii} = 0 \\
& \sum_{j=1}^{n} z_{ij} = K \\
& \mathbf{z}_i \in \{0,1\}^n.
\end{aligned} \tag{2.9}$$

The optimal solution to Problem (2.9) can be found using a simple sorting procedure on the distances $\{d_{ij}\}_{j=1}^{n}$. For each $i \in \mathcal{I}$, we find the K-nearest neighbors of $(\mathbf{w}_i, \mathbf{v}_i)$ and set $z_{ij} = 1$ for these neighbors, $z_{ij} = 0$, otherwise.

Next, we fix $\mathbf{Z}$ and update the imputed values $\mathbf{W}, \mathbf{V}$ using either BCD or CD. In step (2a), the BCD update, Problem (2.3) decomposes by dimension $d = 1, \ldots, p$. For each continuous dimension $d = 1, \ldots, p_0$, we consider the following quadratic optimization problem:

$$\begin{aligned} \min_{\mathbf{w}^d} \quad & \sum_{i \in \mathcal{I}} \sum_{j=1}^{n} z_{ij}(w_{id} - w_{jd})^2 \\ \text{s.t.} \quad & w_{id} = x_{id}, \qquad\qquad (i,d) \in \mathcal{N}_0, \end{aligned} \tag{2.10}$$

where $\mathbf{w}^d \in \mathbb{R}^n$ are the imputed values in the dth dimension. Taking partial derivative of the objective function with respect to w_{id} for some missing entry $(i,d) \in \mathcal{M}_0$ and setting it to zero, we obtain after some simplifications:

$$(K + \sum_{j \in \mathcal{I}} z_{ji}) w_{id} - \sum_{(j,d) \in \mathcal{M}_0} (z_{ij} + z_{ji}) w_{jd} - \sum_{(j,d) \in \mathcal{N}_0} (z_{ij} + \mathbb{1}_{\{j \in \mathcal{I}\}} z_{ji}) x_{jd} = 0. \tag{2.11}$$

For each continuous dimension d, we have a system of equations of the form (2.11) which we can solve to determine the optimal imputed values $w_{id}, (i,d) \in \mathcal{M}_0$. To simplify notation, suppose that the missing values for dimension d are $\widetilde{\mathbf{w}} := (\widetilde{w}_{1d}, \ldots, \widetilde{w}_{ad})$ and the known values are $\widetilde{\mathbf{x}} := (\widetilde{x}_{(a+1)d}, \ldots, \widetilde{x}_{nd})$. Then, the set of optimal imputed missing values $\widetilde{\mathbf{w}}$ is the solution to the linear system $\mathbf{Q}\widetilde{\mathbf{w}} = \mathbf{R}\widetilde{\mathbf{x}}$, where

$$\mathbf{Q} = \begin{bmatrix} K + \sum_{j \in \mathcal{I}} z_{j1} - 2z_{11} & -z_{12} - z_{21} & \cdots & -z_{1a} - z_{a1} \\ -z_{21} - z_{12} & K + \sum_{j \in \mathcal{I}} z_{j2} - 2z_{22} & \cdots & -z_{2a} - z_{a2} \\ \vdots & & \ddots & \vdots \\ -z_{a1} - z_{1a} & -z_{a2} - z_{2a} & \cdots & K + \sum_{j \in \mathcal{I}} z_{ja} - 2z_{aa} \end{bmatrix},$$

$$\mathbf{R} = \begin{bmatrix} z_{1(a+1)} + \mathbb{1}_{\{(a+1)\in\mathcal{I}\}} z_{(a+1)1} & \dots & z_{1n} + \mathbb{1}_{\{n\in\mathcal{I}\}} z_{n1} \\ \vdots & & \vdots \\ z_{a(a+1)} + \mathbb{1}_{\{(a+1)\in\mathcal{I}\}} z_{(a+1)a} & \dots & z_{an} + \mathbb{1}_{\{n\in\mathcal{I}\}} z_{na} \end{bmatrix}.$$

Note that when K is sufficiently large, the matrix $\mathbf{Q}$ is positive semidefinite and therefore invertible. If $\mathbf{Q}$ is singular, then we may add a small positive perturbation to the diagonal of $\mathbf{Q}$ so that the matrix becomes positive semidefinite. Therefore, without loss of generality there is a closed-form solution $\widetilde{\mathbf{w}} = \mathbf{Q}^{-1}\mathbf{R}\widetilde{\mathbf{x}}$ to this system of equations for each continuous dimension d.

In order to update $\mathbf{V}$, we solve the following integer linear optimization problem for each categorical dimension $d = (p_0 + 1), \dots, p$:

$$\begin{aligned} \min_{\mathbf{v}^d} \quad & \sum_{i\in\mathcal{I}} \sum_{j=1}^{n} z_{ij} y_{ij} \\ \text{s.t.} \quad & v_{id} = x_{id} && (i,d) \in \mathcal{N}_1, \\ & v_{id} - v_{jd} \le y_{ij} k_d && i = 1,\dots,n, j = 1,\dots,n, \\ & v_{jd} - v_{id} \le y_{ij} k_d && i = 1,\dots,n, j = 1,\dots,n, \\ & y_{ij} \in \{0,1\}^{n\times n}, \end{aligned} \tag{2.12}$$

where $\mathbf{v}^d \in \{1,\dots,k_d\}^n$ are the imputed values for the dth dimension. Here, the indicator variables y_{ij} take values equal to $\mathbb{1}_{\{v_{jd}\neq v_{jd}\}}$ in the optimal solution.

In step (2b), following the CD method, we update the missing imputed values one at a time. Each $w_{id}, (i,d) \in \mathcal{M}_0$ is imputed as the minimizer of the following:

$$\min_{w_{id}} \quad \sum_{j=1}^{n} z_{ij}(w_{id} - w_{jd})^2 + \sum_{j\in\mathcal{I}} z_{ji}(w_{jd} - w_{id})^2. \tag{2.13}$$

Solving the above gives

$$w_{id} = \frac{\sum_{j=1}^{n} z_{ij} w_{jd} + \sum_{j\in\mathcal{I}} z_{ji} w_{jd}}{K + \sum_{j\in\mathcal{I}} z_{ji}}. \tag{2.14}$$

We can interpret the missing value imputation (2.14) as a weighted average of the K nearest neighbors of $\mathbf{x}_i$, along with all points $\mathbf{x}_j$ which include $\mathbf{x}_i$ as a neighbor. Similarly, each categorical variable $v_{id}, (i,d) \in \mathcal{M}_1$ is imputed as the minimizer of the following:

$$\min_{v_{id}} \quad \sum_{j=1}^{n} z_{ij} \mathbb{1}_{\{v_{id} \neq v_{jd}\}} + \sum_{j \in \mathcal{I}} z_{ji} \mathbb{1}_{\{v_{jd} \neq v_{jd}\}}. \tag{2.15}$$

The solution is

$$v_{id} = \text{mode}\big(\big\{\{v_{jd} : z_{ij} = 1\}, \{v_{jd} : z_{ji} = 1\}\big\}\big). \tag{2.16}$$

Here, we set v_{id} to be the highest frequency category among the K nearest neighbors of $\mathbf{x}_i$, along with all points $\mathbf{x}_j$ which include $\mathbf{x}_i$ as a nearest neighbor. In practice, we use this update for $v_{id}, (i,d) \in \mathcal{M}_1$ in place of the update for $\mathbf{V}$ in BCD because it is much faster computationally.

2.2.4 Mixed SVM Based Imputation

In this section, we consider a second model for imputation, based upon SVM regression for imputing continuous features and SVM classification for imputing categorical features. First, define $\widetilde{\mathbf{v}}_i \in \{-1, 1\}^{p_2}$ to be a dummy encoded representation of $\mathbf{v}_i$, where $p_2 = \sum_{d=p_0+1}^{p_0+p_1} k_d - p_1$. Let $\widetilde{v}_{id}^{fixed}, (i,d) \in \mathcal{N}_2$ be the known dummy encoded values. For each continuous feature $d \in \{1, \ldots, p_0\}$, let $(\boldsymbol{\beta}_d, \beta_{d0}) \in \mathbb{R}^{p_0+p_2+1}$ be the coefficients for an SVM regression model regressing feature d on the other features with the dummy encoding. Let $(\boldsymbol{\theta}_d, \theta_{d0}) \in \mathbb{R}^{p_0+p_2+1}$ be the coefficients for an SVM classification model predicting dummy feature d based upon the other features. Note that it is also possible to use a multi-class SVM model to predict each categorical feature directly, as described in [38] with the parameters $\mathbf{M} \in \mathbb{R}^{k_d \times (p_0+p_2+1)}$ for each feature $d \in \{p_0+1, \ldots, p_0+p_1\}$. In this case, we would keep the dummy encoded decision variables as covariates to predict the other features and add constraints relating $v_{id}, (i,d) \in \mathcal{M}_1$ and $\widetilde{v}_{id}, (i,d) \in \mathcal{M}_2$. For illustrative purposes and simplicity of notation, we present the formulation using binary SVM to predict each dummy variable d.

We consider the following optimization problem:

$$
\begin{aligned}
\min \quad & c([\boldsymbol{\beta},\boldsymbol{\theta}],\mathbf{W},\widetilde{\mathbf{V}};\mathbf{X}) := \frac{1}{2}\left(\|\boldsymbol{\theta}\|^2+\|\boldsymbol{\beta}\|^2\right) \\
& \qquad\qquad +C\left(\sum_{i=1}^{n}\sum_{d=1}^{p_0}(\gamma_{id}+\gamma^*_{id})+\sum_{i=1}^{n}\sum_{d=p_0+1}^{p_0+p_1}\xi_{id}\right) \\
\text{s.t.} \quad & x_{id} = w_{id}- && (i,d)\in\mathcal{N}_0, \\
& \widetilde{v}_{id} = \widetilde{v}^{fixed}_{id} && (i,d)\in\mathcal{N}_2, \\
& \beta_{dd} = 0 && d=1,\ldots,p_0, \\
& \theta_{dd} = 0 && d=1,\ldots,p_2, \\
& \gamma_{id} \geq w_{id} - (\boldsymbol{\beta}_d^T \begin{bmatrix}\mathbf{w}_i\\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \beta_{d0}) - \epsilon && d=1,\ldots,p_0, i=1\ldots,n, \\
& \gamma^*_{id} \geq (\boldsymbol{\beta}_d^T \begin{bmatrix}\mathbf{w}_i\\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon && d=1,\ldots,p_0, i=1\ldots,n, \\
& \xi_{id} \geq 1 - \widetilde{v}_{id}(\boldsymbol{\theta}_d^T \begin{bmatrix}\mathbf{w}_i\\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \theta_{d0}) && d=1,\ldots,p_2, i=1\ldots,n, \\
& \gamma_{id} \geq 0 && d=1,\ldots,p_0, i=1\ldots,n, \\
& \gamma^*_{id} \geq 0 && d=1,\ldots,p_0, i=1\ldots,n, \\
& \xi_{id} \geq 0 && d=1,\ldots,p_2, i=1\ldots,n, \\
& \widetilde{v}_{id} \in \{-1,1\} && d=1,\ldots,p_2, i=1\ldots,n.
\end{aligned}
\tag{2.17}
$$

This formulation is based upon SVM with a linear kernel; however we can extend Problem (2.17) to arbitrary kernels, including the multi-class cases, using the modified objective function

$$
c([\boldsymbol{\beta},\boldsymbol{\theta}],\mathbf{W},\mathbf{V};\mathbf{X}) := \frac{1}{2}(\|\boldsymbol{\beta}\|^2_{\mathcal{H}}+\|\boldsymbol{\theta}\|^2_{\mathcal{H}}) + C\left(\sum_{i=1}^{n}\sum_{d=1}^{p_0}(\gamma_{id}+\gamma^*_{id})+\sum_{i=1}^{n}\sum_{d=p_0+1}^{p_0+p_1}\xi_{id}\right), \tag{2.18}
$$

where $\|\cdot\|_{\mathcal{H}}$ is the norm in a given Reproducing Kernel Hilbert Space $\mathcal{H}$.

Another important aspect of Problem (2.17) is the compound objective function, which is the summation of objective functions derived from both SVM regression and

SVM classification methods. Observe that if we fix a single imputed entry w_{id} or $\widetilde{v}_{id}$, the contribution to the objective function scales linearly as $(\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0})$ if d is continuous or scales linearly as $(\boldsymbol{\theta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \theta_{d0})$ if d is categorical. This is desirable because we do not wish to weight continuous and categorical variables unequally in our imputation. Next, we describe the updates in Algorithm 1 for mixed SVM based imputation, which we refer to as `opt.svm`.

In step ①, we fix the imputed values $\mathbf{W}, \mathbf{V}$ and update the auxiliary variables $[\boldsymbol{\beta}, \boldsymbol{\beta}_0, \boldsymbol{\theta}, \boldsymbol{\theta}_0]$. Independent of the choice of kernel, Problem (2.2) decomposes by dimension p into p_0 SVM regression problems and p_2 SVM classification problems for the categorical variables. For each continuous feature $d \in \{1, \ldots, p_0\}$, we update $\boldsymbol{\beta}_d, \beta_{d0}$ by solving

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}(\gamma_{id} + \gamma_{id}^*) \\
\text{s.t.} \quad & \beta_{dd} = 0 \\
& \gamma_{id} \geq w_{id} - (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - \epsilon \quad i = 1 \ldots, n, \\
& \gamma_{id}^* \geq (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon \quad i = 1 \ldots, n, \\
& \gamma_{id} \geq 0 \qquad\qquad\qquad\qquad\qquad\quad i = 1, \ldots, n, \\
& \gamma_{id}^* \geq 0 \qquad\qquad\qquad\qquad\qquad\quad i = 1, \ldots, n.
\end{aligned}
\tag{2.19}
$$

Similarly, for each dummy feature $d \in \{p_0 + 1, \ldots, p_0 + p_2\}$, we update $\boldsymbol{\theta}_d, \theta_{d0}$ by

solving

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{i=1}^{n}\xi_{id} \\ \text{s.t.} \quad & \theta_{dd} = 0 \\ & \xi_{id} \geq 1 - \widetilde{v}_{id}(\boldsymbol{\theta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \theta_{d0}) \quad i = 1\ldots, n, \\ & \xi_{id} \geq 0 \qquad\qquad\qquad\qquad\qquad i = 1, \ldots, n. \end{aligned} \tag{2.20}$$

Taking the Lagrangian duals, both Problems (2.19) and (2.20) can be reformulated as quadratic optimization problems which can be solved efficiently [36].

Next, we fix the auxiliary variables $[\boldsymbol{\beta}, \boldsymbol{\beta}_0, \boldsymbol{\theta}, \boldsymbol{\theta}_0]$ and update the imputed values $\mathbf{W}, \mathbf{V}$ using BCD or CD. In step (2a), Problem (2.2) decomposes by observation i into n nonlinear integer optimization problems. For each i we solve

$$\begin{aligned} \min_{\mathbf{w}_i, \widetilde{\mathbf{v}}_i} \quad & \sum_{d=1}^{p_0}(\gamma_{id} + \gamma_{id}^*) + \sum_{d=p_0+1}^{p_0+p_1}\xi_{id} \\ \text{s.t.} \quad & x_{id} = w_{id} && (i,d) \in \mathcal{N}_0, \\ & \gamma_{id} \geq w_{id} - (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - \epsilon && d = 1, \ldots, p_0, \\ & \gamma_{id}^* \geq (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon && d = 1, \ldots, p_0, \\ & \xi_{id} \geq 1 - \widetilde{v}_{id}(\boldsymbol{\theta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \theta_{d0}) && d = 1, \ldots, p_2, \\ & \gamma_{id} \geq 0 && d = 1, \ldots, p_0, \\ & \gamma_{id}^* \geq 0 && d = 1, \ldots, p_0, \\ & \xi_{id} \geq 0 && d = 1, \ldots, p_2, \end{aligned} \tag{2.21}$$

where $(\mathbf{w}_i, \widetilde{\mathbf{v}}_i) \in \mathbb{R}^{p_0} \times \{-1, 1\}^{p_2}$ is the imputation for observation $\mathbf{x}_i$. Note that if all features are continuous, Problem (2.21) reduces to a linear optimization problem. Because we are using the dummy encoding in this formulation, it is possible to obtain

an imputation in which multiple classes are selected for a single categorical entry. In this case, when `opt.svm` terminates, we select the imputation among the set of potential candidates which minimizes the objective function of Problem (2.21).

In step (2b), we update the imputed values one at a time. To update $w_{id}, (i,d) \in \mathcal{M}_0$, we solve the one-dimensional linear optimization problem:

$$\begin{aligned}
\min_{w_{id}} \quad & \sum_{d=1}^{p_0}(\gamma_{id}+\gamma_{id}^*) + \sum_{d=p_0+1}^{p_0+p_1}\xi_{id} \\
\text{s.t.} \quad & \gamma_{id} \geq w_{id} - (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - \epsilon \quad d = 1,\ldots,p_0, \\
& \gamma_{id}^* \geq (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon \quad d = 1,\ldots,p_0, \\
& \xi_{id} \geq 1 - \widetilde{v}_{id}(\boldsymbol{\theta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \theta_{d0}) \qquad d = 1,\ldots,p_2, \\
& \gamma_{id} \geq 0 \qquad\qquad\qquad\qquad d = 1,\ldots,p_0, \\
& \gamma_{id}^* \geq 0 \qquad\qquad\qquad\qquad d = 1,\ldots,p_0, \\
& \xi_{id} \geq 0 \qquad\qquad\qquad\qquad d = 1,\ldots,p_2.
\end{aligned} \tag{2.22}$$

We update $\widetilde{v}_{id}, (i,d) \notin \mathcal{N}_2$ by solving the binary optimization problem:

$$\begin{aligned}
\min_{\widetilde{v}_{id}\in\{-1,1\}} \quad & \sum_{i=1}^{n}\sum_{d=1}^{p_0} \big(\max\{w_{id} - (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - \epsilon, 0\} + \\
& \max\{(\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon, 0\}\big) + \\
& \sum_{i=1}^{n}\sum_{d=1}^{p_2} \big(1 - \widetilde{v}_{id}(\boldsymbol{\theta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \theta_{d0})\big).
\end{aligned} \tag{2.23}$$

2.2.5 Tree Based Imputation

Finally, we consider an imputation model based on classification and regression trees. For each dimension we train a decision tree to predict the missing values, using the other features as covariates. We train regression trees to predict each of the continuous variables and classification trees to predict each of the categorical variables. Given a regression tree for continuous dimension d, we will impute $x_{id}, (i,d) \in \mathcal{M}_0$ to be the mean in dimension d of all points in the same leaf node as $\mathbf{x}_i$. Similarly, given a classification tree for dimension d, we will impute $x_{id}, (i,d) \in \mathcal{M}_1$ to be the mode in dimension d of all points in the same leaf node as $\mathbf{x}_i$.

For general prediction tasks, we can use greedy [29] or globally optimal [7] solution methods to train the decision trees. In this case, we consider the latter approach because it admits a clear optimization model with mixed integer decision variables which fits into our framework for imputation. For each dimension d, let $\mathbf{T}^d \in \{0,1\}^{n \times n}$ denote the set of indicator variables

$$t_{ij}^d = \begin{cases} 1, & \text{if } (\mathbf{w}_i, \mathbf{v}_i), (\mathbf{w}_j, \mathbf{v}_j) \text{ are in the same leaf node} \\ & \text{of the decision tree for dimension } d, \\ 0, & \text{otherwise.} \end{cases} \tag{2.24}$$

Let $(\mathbf{T}^d, \mathbf{W}, \mathbf{V}) \in \mathcal{T}^d$ denote the set of optimal decision tree constraints for dimension d as described in [7]. We consider the following optimization problem:

$$\begin{aligned} \min \quad & c(\mathbf{T}, \mathbf{W}, \mathbf{V}; \mathbf{X}) := \sum_{i=1}^{n} \sum_{j=1}^{n} \Big[\sum_{d=1}^{p_0} \; t_{ij}^d (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} t_{ij}^d \mathbb{1}_{\{v_{id} \neq v_{jd}\}} \Big] \\ \text{s.t.} \quad & w_{id} = x_{id} && (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} && (i,d) \in \mathcal{N}_1, \\ & (\mathbf{T}^d, \mathbf{W}, \mathbf{V}) \in \mathcal{T}^d, && d = 1, \ldots, p, \end{aligned} \tag{2.25}$$

Next, we describe the updates in Algorithm 1 for decision tree based imputation, which we refer to as `opt.tree`.

In step ①, we fix the imputed values $\mathbf{W}, \mathbf{V}$ and update the decision tree variables $\mathbf{T}$. For each continuous feature, we fit a regression tree to predict $\mathbf{w}^d$ based upon the other features. Similarly, for each categorical feature, we fit a classification tree to predict $\mathbf{v}^d$ based upon the other features. In practice, we may use greedy or optimal methods to find these trees; however, if we use greedy trees then the objective function value $c(\mathbf{T}, \mathbf{W}, \mathbf{V}; \mathbf{X})$ is not guaranteed to be monotonically decreasing over the course of the algorithm.

Next, we fix $\mathbf{T}$ and update the imputed values $\mathbf{W}, \mathbf{V}$ using BCD or CD. In step (2a), Problem (2.3) decomposes by dimension into p_0 quadratic optimization problems and p_1 integer optimization problems. For each continuous dimension $d = 1, \ldots, p_0$, we solve:

$$\begin{aligned} \min_{\mathbf{w}^d} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij}^d (w_{id} - w_{jd})^2 \\ \text{s.t.} \quad & w_{id} = x_{id}, \qquad\qquad (i,d) \in \mathcal{N}_0, \end{aligned} \tag{2.26}$$

where $\mathbf{w}^d \in \mathbb{R}^n$ are the imputed values in the dth dimension. This is a quadratic optimization problem with an explicit optimum. For each $w_{id}, (i,d) \in \mathcal{M}_0$, an optimal solution is

$$w_{id} = \begin{cases} \dfrac{\sum_{(j,d) \in \mathcal{N}_0^d} t_{ij}^d x_{jd}}{\sum_{(j,d) \in \mathcal{N}_0^d} t_{ij}^d}, & \text{if} \quad \sum_{(j,d) \in \mathcal{N}_0^d} t_{ij}^d \geq 1, \\ \dfrac{1}{|\mathcal{N}_0^d|} \displaystyle\sum_{(j,d) \in \mathcal{N}_0^d} x_{jd}, & \text{otherwise}, \end{cases} \tag{2.27}$$

where $\mathcal{N}_0^d := \{(i,r) \in \mathcal{N}_0 : r = d\}$. This solution corresponds to setting each missing entry equal to the mean of all observed values in the same leaf node. If the number of non-missing values in the same leaf node as w_{id} is zero, i.e., $\sum_{(j,d) \in \mathcal{N}_0^d} t_{ij}^d = 0$, then we set all of the values in that leaf node to the mean impute solution.

For each categorical dimension $d = p_0 + 1, \ldots, p_0 + p_1$, we solve the following integer optimization problem:

$$\begin{aligned} \min_{\mathbf{v}^d} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij}^d \mathbb{1}_{\{v_{id} \neq v_{jd}\}} \\ \text{s.t.} \quad & v_{id} = x_{id}, \qquad\qquad (i,d) \in \mathcal{N}_1, \end{aligned} \tag{2.28}$$

where $\mathbf{v}^d \in \{1, \ldots, k_d\}^n$ are the imputed values for the dth dimension. An optimal solution is

$$v_{id} = \begin{cases} \text{mode}\big(\{x_{jd} : t_{ij}^d = 1, (j,d) \in \mathcal{N}_1\}\big) & \text{if} \quad |\{x_{jd} : t_{ij}^d = 1, (j,d) \in \mathcal{N}_1\}| \geq 1, \\ \text{mode}\big(\{x_{jd} : (j,d) \in \mathcal{N}_1\}\big) & \text{otherwise.} \end{cases} \tag{2.29}$$

In step (2b), we update the missing imputed values one at a time, which results in slightly different closed form solutions for $w_{id}, (i,d) \in \mathcal{M}_0$ and $v_{id}, (i,d) \in \mathcal{M}_1$. First, we update the continuous variables $w_{id}, (i,d) \in \mathcal{M}_0$ by solving:

$$\min_{w_{id}} \quad 2\sum_{j=1}^{n} t_{ij}^d (w_{id} - w_{jd})^2. \tag{2.30}$$

An optimal solution to Problem (2.30) is

$$w_{id} = \begin{cases} \dfrac{\sum_{j \neq i} t_{ij}^d w_{jd}}{\sum_{j \neq i} t_{ij}^d}, & \text{if} \quad \sum_{j \neq i} t_{ij}^d \geq 1, \\ \dfrac{1}{|\mathcal{N}_0^d|} \displaystyle\sum_{(j,d) \in \mathcal{N}_0^d} x_{jd}, & \text{otherwise.} \end{cases} \tag{2.31}$$

Next, we update the categorical variables $v_{id}, (i,d) \in \mathcal{M}_1$ one at a time by solving:

$$\min_{v_{id}} \quad 2\sum_{j=1}^{n} t_{ij}^d \mathbb{1}_{\{v_{id} \neq v_{jd}\}}, \tag{2.32}$$

An optimal solution to Problem (2.32) is

$$v_{id} = \begin{cases} \text{mode}\big(\{v_{jd} : t_{ij}^d = 1\}\big), & \text{if} \quad |\{v_{jd} : t_{ij}^d = 1\}| \geq 1, \\ \text{mode}\big(\{x_{jd} : (j,d) \in \mathcal{N}_1\}\big), & \text{otherwise.} \end{cases} \tag{2.33}$$

Both of these updates coincide with the predicted values from the decision trees constructed.

2.2.6 Model Selection Procedure

Each of the above methods and choice of hyperparameters generates some imputed values. For single imputation, a single set of imputed values should be generated in the end. We propose the following procedure for model selection.

Given $\mathbf{X}$ with existing missing data $\mathcal{M}_0, \mathcal{M}_1$, we generate an additional fixed percentage of data missing $\mathcal{M}_0^{valid}, \mathcal{M}_1^{valid}$, with the known values as the hold-out set, and perform each of the imputation methods under the combined missing pattern. We evaluate the imputation quality on the hold-out validation set by measuring how closely the imputed values resemble the ground truth values. In particular, the mean absolute error (MAE) between true and imputed values for each imputation method is calculated. The validation MAE is defined to be

$$\frac{1}{|\mathcal{M}_0^{valid}|} \sum_{(i,d)\in\mathcal{M}_0^{valid}} |w_{id} - x_{id}| + \frac{1}{|\mathcal{M}_1^{valid}|} \sum_{(i,d)\in\mathcal{M}_1^{valid}} \mathbb{1}_{\{v_{id}\neq x_{id}\}}. \tag{2.34}$$

Lower values indicate closer imputation, and perfect imputation corresponds to an MAE of zero. Another metric of imputation quality is root mean squared error (RSME), which is given by

$$\sqrt{\frac{1}{|\mathcal{M}_0^{valid}|} \sum_{(i,d)\in\mathcal{M}_0^{valid}} (w_{id} - x_{id})^2 + \frac{1}{|\mathcal{M}_1^{valid}|} \sum_{(i,d)\in\mathcal{M}_1^{valid}} \mathbb{1}_{\{v_{id}\neq x_{id}\}}}. \tag{2.35}$$

For each imputation method, the combination of hyperparameters that achieves the lowest MAE in validation (or RMSE) is selected, and the $\mathbf{X}$ is again imputed but under the original missing patterns $\mathcal{M}_0$, $\mathcal{M}_1$. This set of imputed values is now ready to be evaluated or used for downstream tasks.

The hyperparameters that we tune via this method are summarized in Table 2.3. In addition, we also use this cross-validation procedure to select the best method out of `opt.knn`, `opt.svm`, and `opt.tree`. We refer to this composite method as `opt.cv`. Similarly, we may use the cross-validation procedure for model selection for any set of imputations. We define `benchmark.cv` to be the procedure that selects

Table 2.3: Hyperparameters tuned via the model selection procedure outlined in Section 2.2.6. σ^2 is a parameter in the radial basis function kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma^2})$. cp is a complexity parameter related to the depth of the decision tree.

Method	**Hyperparameters**
K-NN	K
SVM	C, σ^2
Trees	cp

the best method out of: `mean`, `pmm`, `bpca`, `knn`, and `iknn` that will be later used in computational comparisons (see Section 2.3.1 for descriptions of these individual methods).

2.2.7 Extensions to Multiple Imputation

Thus far, we have described `opt.impute` methods for single imputation which output a single completed data set. On the other hand, multiple imputation methods output $m \geq 2$ different completed data sets for a single missing data problem. Afterwards, analysis is performed on each of the m data sets separately, and the results are pooled [69]. For some applications, multiple imputation is preferred because it captures the variation in missing data imputation, which enables us to compute confidence intervals for downstream models trained on the imputed data sets. In addition, the pooled results from models fit on multiple imputed data sets may provide better point estimates than models fit on a single imputed data set in some cases.

To extend `opt.impute` to produce multiple imputations, we generate m warm starts using a probabilistic procedure, run `opt.knn`, `opt.svm`, or `opt.tree` from these starting points, and output the full set of m completed data sets. These warm starts can be generated from sample draws under a previously estimated posterior distribution; an example would be using outputs from the `mice` procedure. This provides us with a representative set of imputations found by the `opt.impute` algorithm, which converges to local optima. We refer to the multiple imputation method as `opt.mi`. In the computational experiments, we use the benchmark multiple imputation method

`mice` to generate the warm starts.

Note that there are other possible ways of adapting `opt.impute` to the multiple imputation schema. We may introduce m instances of artificial noise in the observed values, and solve the resulting optimization problems. Alternatively, we may run `opt.impute` on m bootstrapped samples of the original data set. Afterwards, we can analyze each of the m imputed data sets separately and pool the results as before.

2.3 Real-World Data Experiments

In this section, we evaluate the performance of `opt.impute` on many real-world data sets. Our comparisons include 1) the effect on imputation accuracy, and 2) the effect on the performance of downstream machine learning tasks. We compare to the most commonly used state-of-the-art methods on a large sample of data sets from the UCI Machine Learning Repository. For data sets that include categorical variables, we impute the discrete values directly using our specialized imputation methods for categorical variables and benchmark methods.

2.3.1 Experimental Setup

To test the accuracy of the proposed missing data imputation method, we run a series of computational experiments on data sets taken from the UCI Machine Learning Repository for both regression and classification tasks. The data sets cover a range of number of observations n and number of features p, potentially mixed with both continuous and categorical variables. The numbers of continuous (p_0) and categorical (p_1) variables in each of these data sets are given in Table 2.10.

In these experiments, we use full data sets in which all entries are known, and we generate patterns of missing data for various percentages ranging from 10% to 50%. We take the full data sets $\mathbf{X}$ that have no missing entries to be the ground truth. We run some of the most commonly-used and state-of-the-art methods for data imputation on these data sets to predict the missing values and compare against our optimization based imputation methods. The individual methods in this comparison

are:

1. **Mean Impute** (`mean`): The simplest imputation method. For each missing value x_{id}, imputes the mean of all known values in dimension d.

2. **Predictive-Mean Matching** (`pmm`): An iterative method which imputes missing values from known values in a given dimension using linear regressions. It is commonly used for multiple imputation and can be generalized to multiple missing dimensions using the chained equations process [32]. Implemented using the `MICE` package in R.

3. **Bayesian PCA** (`bpca`): A missing data estimation method based on Bayesian principal component analysis [76]. Implemented using the `pcaMethods` package in R.

4. K-**Nearest Neighbors** (`knn`): A single-step, greedy method which imputes missing values using the K-nearest neighbors of an observation based upon Euclidean distance. The candidate neighbors must have non-missing values in the imputed feature. Averaged distance is used if some other coordinates are missing. Implemented using the `impute` package in R.

5. **Iterative** K-**Nearest Neighbors** (`iknn`): Implemented in R and Julia, based on the description in the original papers [27, 34] .

6. **Optimal Impute** (`opt.impute`): All sub-methods below use warm starts including: `mean`, `knn`, `bpca` and five random starts where the values are imputed by a random sampling of the non-missing observations of that feature. The imputation which results in the lowest objective value is selected for each method.

   (a) K-NN based (`opt.knn`): This method solves the optimal K-nearest neighbors problem (2.8). Convergence time depends upon the quality of the initial warm start. We run both block coordinate descent and coordinate descent for small data sets of size $n \leq 10{,}000$, and only coordinate descent for large data sets with higher n. The implementation was in the

programming language Julia with fast algorithms for K-nearest neighbor calculations.

(b) SVM Regression and Classification based (`opt.svm`): This method solves the maximum margin support vector machine problem (2.17) using a radial basis function kernel. For continuous variables, we use ϵ-support vector regression; for categorical variables, we use classical support vector machines. These problems were solved using coordinate descent methods. The implementation was in Julia using the `scikit-learn` package in Python.

(c) Decision Tree based (`opt.tree`): This method solves the optimal decision-tree problem (2.25). For continuous variables, a single-leaf regularized regression tree is used; for categorical variables, a fast coordinate descent-based algorithm for solving Optimal Classification Trees is used [7]. We run coordinate descent for the imputation problems. The implementation was in Julia using the packages `glmnet` and `OptimalTrees`.

In addition, we consider two composite methods: `opt.cv`, which selects the best method from `opt.knn`, `opt.svm`, and `opt.tree`; and `benchmark.cv`, which selects the best method from `mean`, `pmm`, `bpca`, `knn`, and `iknn`. These composite methods use the cross-validation procedure described in Section 2.2.6. To generate the validation set for each missing data problem, we randomly sample an additional 10% of the entries to be hidden under the MCAR assumption. After running each individual method, we select the one that gives the lowest MAE on the validation set. We re-run this method on the original missing data set to obtain the final imputation.

Each imputation method was run for a maximum time limit of 12 hours on each data set. The quality of the imputations is evaluated using the same MAE and RMSE metrics defined in Section 2.2.6. For each of the `opt.impute` methods, we also record and present the convergence in objective value and MAE to show the progress over the iterations.

Table 2.4: Statistical assumptions of mechanisms used to generate patterns of missing data $\mathcal{M}$ for data set $\mathbf{X}$. Here, we suppose that f is the underlying density of the missing pattern, and $\mathbf{X}^{obs}, \mathbf{X}^{miss}$ are the observed and missing components of the data set, respectively.

Mechanism of Missing Data	**Assumption**
Missing Completely at Random (MCAR)	$f(\mathcal{M}\|\mathbf{X}^{obs}, \mathbf{X}^{miss}) = f(\mathcal{M})$
Missing at Random (MAR)	$f(\mathcal{M}\|\mathbf{X}^{obs}, \mathbf{X}^{miss}) = f(\mathcal{M}\|\mathbf{X}^{obs})$
Not Missing at Random (NMAR)	$f(\mathcal{M}\|\mathbf{X}^{obs}, \mathbf{X}^{miss})$ is a function of $\mathbf{X}^{miss}$

Missing Pattern

Because the mechanism which generates the pattern of missing data can affect imputation quality, we run experiments under two different missing data mechanisms: missing completely at random (MCAR) and not missing at random (NMAR). These statistical assumptions are summarized in Table 2.4. The MCAR assumption implies that the missing pattern is completely independent from both the missing and observed values. The NMAR assumption implies that the missing pattern depends upon the missing values. There is an intermediate type of assumption, missing at random (MAR), which implies that the missing pattern depends only upon the observed values, but not upon the missing values. Because this assumption is less general than NMAR, we do not consider this mechanism for our experiments.

To generate MCAR patterns of missing data, we randomly sample a subset of the entries in $\mathbf{X}$ to be missing, assuming that each entry is equally likely to be chosen. The NMAR patterns are generated by sampling missingness indicators as independent Bernoulli random variables where each probability p_{id} equals the probability that a normal random variable $N(x_{id}, \epsilon)$ is greater than a particular threshold for dimension d. The threshold for each dimension d is the quantile of $\mathbf{X}^d$ which corresponds to the desired missing percentage level.

Note that regardless of the missing data scenarios generated for the experiments, in order to make fair comparisons, we always use MCAR as the generating mechanism for cross validation.

Downstream Tasks

For 10 data sets from the UCI Machine Learning Repository, we run further experiments to evaluate the impact of these imputations on the intended downstream machine learning tasks. This selection includes a representative sample of 5 data sets for regression and 5 data sets for classification, with dependent variable observations $\mathbf{Y} \in \mathbb{R}^n$ and $\mathbf{Y} \in \{0, 1\}^n$ respectively. We evaluate both single and multiple imputation methods in these experiments.

For single imputation, we consider `opt.cv` and `benchmark.cv`. First, we divide each downstream data set using a 50% training/testing split. Next, we randomly sample a fixed percentage of the entries in $\mathbf{X}$ to be missing completely at random, ranging from 10% to 50%. For each missing percentage, we impute the missing values in the training set and then fit standard machine learning algorithms to obtain a classification or regression model. We impute the missing values in the testing set by running the imputation methods on the full data set. For the regression tasks, we fit cross-validated LASSO and SVR models and compute the out-of-sample accuracy on the imputed testing set. For the classification tasks, we fit cross-validated SVM and Optimal Trees models and compute the out-of-sample R^2 on the imputed testing set.

We also evaluate the performance of multiple imputation methods on the downstream tasks. In these experiments, we consider the following methods:

1. **Multivariate Imputation by Chained Equations** (`mice`): An iterative method which imputes each dimension with missing values one at a time drawing from distributions fully conditional on the other variables. We use predictive mean matching for continuous variables and logistic regression for categorical variables. This process is repeated to generate m fully imputed data sets. Implemented via the `MICE` package in `R`.

2. **Optimal Impute for Multiple Imputation** (`opt.mi`): Starting from m warm starts, we run `opt.knn`, `opt.svm`, or `opt.tree` to generate a new set of m fully imputed data sets. We use warm starts produced by `mice`, and the best

model among K-NN, SVM, and trees is selected initially via cross-validation.

For both `mice` and `opt.mi`, we generate $m = 5$ multiple imputations for the training set and fit an ensemble of predictive models on these completed training sets. We make predictions on the test set by averaging the predictions from the model ensemble. For the classification tasks, we use a threshold value of 0.5. We run this experiment 100 times with different training/testing splits and distributions of missing values for each data set and report the averaged out-of-sample of the predictive models.

2.3.2 Results

We run the methods on 95 data sets from the UCI Machine Learning Repository. These data sets range in size from $n = 47$ to 20,000 observations and dimension $p = 3$ to 279. In the following sections, we first show the convergence for each of the `opt.impute` methods is fast and generally leads to a decrease in MAE. Next, we demonstrate that the quality of the imputations is significantly higher for `opt.impute` compared to the reference methods, and that this leads to improved performance on downstream classification and regression tasks. We further discuss the sensitivity of imputation quality to the model parameters (K, cp, C), warm starts, descent method (BCD or CD), and data characteristics including the missing pattern. Finally, we compare the computational burden of each method.

Convergence

Figure 2-1 represents the change in objective value and MAE over the iterations for each of the `opt.impute` methods based on mean warm start, using `iris` data set as an example. We present results for `opt.knn` (CD and BCD), `opt.svm` (CD), and `opt.tree` (CD). The convergence is relatively fast for all methods; in particular, the BCD algorithm for knn converges significantly faster than the CD algorithm. When comparing the change in MAE, the value generally monotonically decreases with each iteration in concordance with the change in objective, especially during the first few iterations. In some paths, MAE increases slightly after a certain point. RMSE

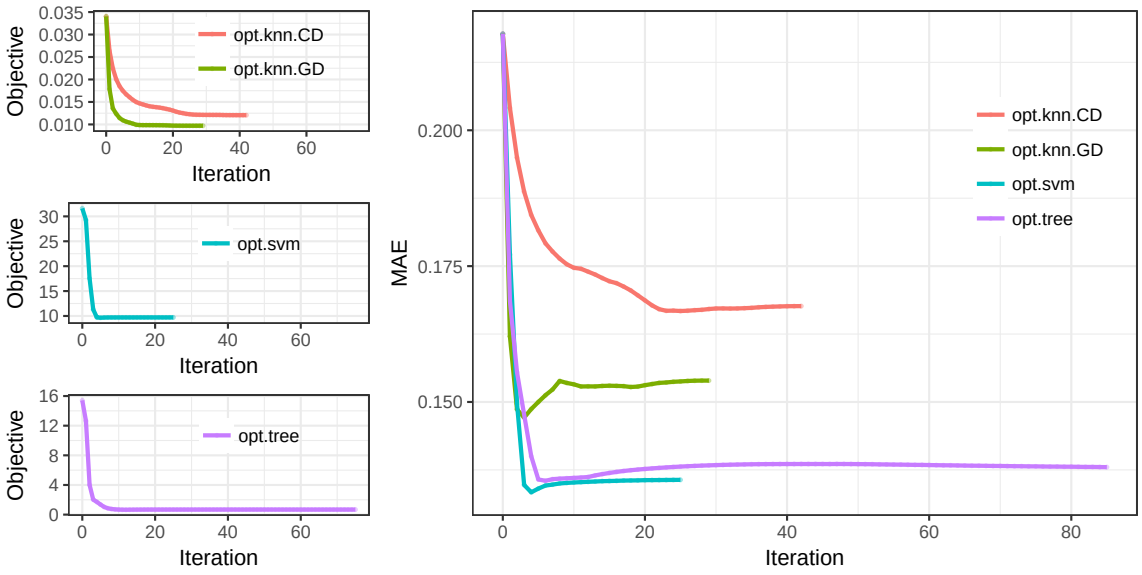


Figure 2-1: Solution paths of `opt.impute` methods on the `iris` data set. These plots show the objective value and mean absolute error (MAE) of the imputation over the course of the algorithm. Each path represents a different algorithm: `opt.knn` (BCD and CD), `opt.svm` (CD), and `opt.tree` (CD). Mean imputation warm start is used.

exhibits the same behavior and is therefore not plotted. This suggests a potential issue of overfitting to the known observations, which may be remedied by regularization or early stopping. In summary, the solution paths illustrate: 1) convergence is often fast, and 2) the objective functions are decent proxies for out-of-sample MAEs, and 3) imputation quality for each first-order method generally improves until convergence.

In general, we found that the BCD algorithm for `opt.knn` did not significantly improve upon imputation accuracy compared to the CD algorithm, but only improved upon speed. Because the BCD algorithms do not scale as well, we restricted our analysis to the CD algorithms for `opt.svm` and `opt.tree`.

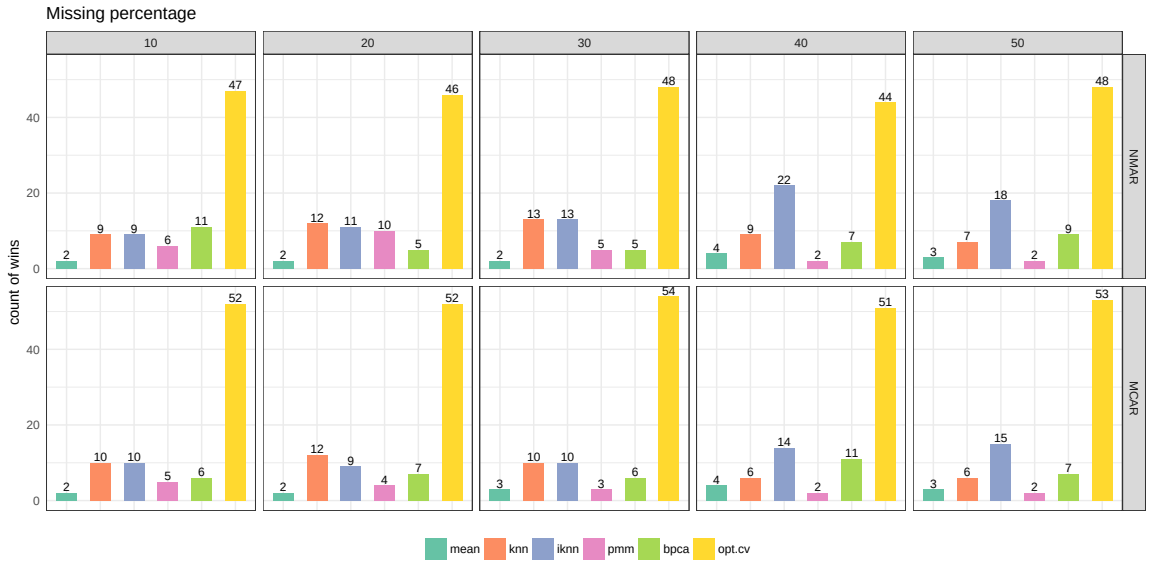
Imputation Accuracy

The imputation accuracy for each data set is presented in Table 2.10 for the scenario in which 30% of the entries are missing, assuming MCAR. We compare the benchmark ones and each individual `opt.impute` method (not cross validated); the method with the lowest MAE (i.e., best imputation accuracy) is bolded. Among all data sets,

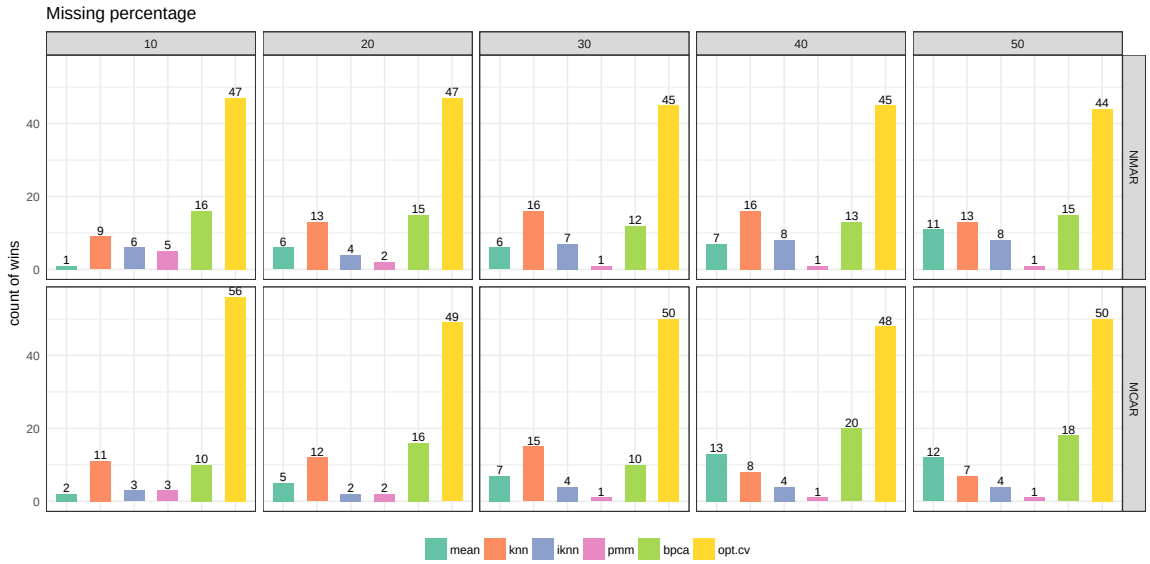
at least one of the `opt.impute` methods obtains the lowest MAE in 76.2% of the data sets, followed by `iknn` and `bpca` imputation methods with 9 and 4 wins each. Comparatively, `mean`, `knn`, and `pmm` impute have the weakest performances. Among the `opt.impute` methods, the tree based model achieves the lowest MAE in most data sets.

We repeat this experiment for other percentages of missing data with the winning counts summarized in Figure 2-2, using `opt.cv` as our proposed method. We show the number of times that each method achieves the best overall imputation with lowest MAE and RMSE under five different missing data percentages, as well MCAR and NMAR scenarios. In all missing data scenarios, our proposed method produces the best imputations in more than half of the data sets according to both performance metrics. Among the comparator methods, `mean` and `pmm` are generally among the weaker ones. When MAE is the metric, the heuristic method `iknn` performs the best among the benchmark methods, suggesting that the idea of iteratively updating the imputed values have merits. At higher percentages of missing values (the right-most subfigures), `bpca` improves in its performance when RMSE is the metric of evaluation, but is still not as strong as `opt.cv`.

In Figure 2-3, we present summary results of the MAE and RMSE values as geometric means across all data sets for each missing percentage and missing data mechanism, with the confidence bands representing one geometric standard deviation multiplied above and divided below by the mean. Comparatively, `opt.cv` achieves the lowest average MAE and RMSE values for all missing percentages. At the 10% missing data percentage, the average MAE of the `opt.cv` imputations is 0.100, a reduction of 14.9% from the average MAE of 0.118 obtained by the best benchmark method `knn`. As missing percentages increase, `opt.cv` remains the most accurate imputation method, with the average MAE of 0.142 at 50% missing, a reduction of 12.1% from the average MAE of 0.172 obtained by the next best method `knn`. The performance of `opt.cv` relative to benchmark ones does not appear to differ drastically between the MCAR and NMAR scenarios, with overall higher MAE for NMAR across most methods, as expected.



(a) Counts based on lowest MAE



(b) Counts based on lowest RMSE

Figure 2-2: Number of data sets in which each missing data imputation method achieves lowest mean absolute error (MAE) and root mean squared error (RMSE) from true value. Each panel represents a different missing percentage ranging from 10% to 50%. Panels in the top row are for not missing at random scenarios, whereas the ones in the bottom row are for missing completely at random scenarios.

Table 2.5: Pairwise Wilcoxon signed-rank tests and t-tests between `opt.impute` and benchmark methods, with the p-values adjusted for multiple comparisons.

`opt.impute`	**Benchmark**	Δ rank (adjusted p-value)	Δ MAE (adjusted p-value)
opt.cv	mean	-0.7855 (<0.001***)	-0.0502 (<0.001***)
	pmm	-0.8355 (<0.001***)	-0.0399 (<0.001***)
	bpca	-0.6329 (<0.001***)	-0.0214 (0.0019**)
	knn	-0.6281 (<0.001***)	-0.0134 (0.0499*)
	iknn	-0.5352 (<0.001***)	-0.0199 (0.0046**)
opt.knn	mean	-0.6424 (<0.001***)	-0.0419 (<0.001***)
	pmm	-0.6091 (<0.001***)	-0.0316 (<0.001***)
	bpca	-0.4875 (<0.001***)	-0.0131 (0.0601)
	knn	-0.3850 (<0.001***)	-0.0051 (0.4574)
	iknn	-0.3611 (<0.001***)	-0.0116 (0.1011)
opt.svm	mean	-0.5852 (<0.001***)	-0.0355 (<0.001***)
	pmm	-0.4875 (<0.001***)	-0.0252 (<0.001***)
	bpca	-0.2515 (<0.001***)	-0.0067 (0.3335)
	knn	-0.1371 (0.0033**)	+0.0013 (0.8485)
	iknn	-0.0322 (0.0884)	-0.0052 (0.4589)
opt.tree	mean	-0.7139 (<0.001***)	-0.0454 (<0.001***)
	pmm	-0.7712 (<0.001***)	-0.0351 (<0.001***)
	bpca	-0.5137 (<0.001***)	-0.0165 (0.0176*)
	knn	-0.4136 (<0.001***)	-0.0086 (0.2152)
	iknn	-0.3135 (<0.001***)	-0.0151 (0.0337*)

To isolate the effect of each individual method from the cross-validation procedure, we further summarize the results by comparing one method at a time against the benchmark ones. Table 2.5 presents the statistical comparisons between each `opt.impute` method and each benchmark method. We conduct pairwise Wilcoxon signed rank tests and paired t-tests between each pair of methods. When comparing `opt.cv` against the benchmark methods, our proposed cross-validated method achieves significantly lower rank and signficantly lower MAE compared to each benchmark one. For each individual `opt.impute` method, with the exception of `opt.svm` against heuristic `iknn`, the `opt.impute` one has significantly lower rank. The decrease in MAE is still significant when `mean`, `bpca`, and `pmm` are comparators, but no longer significant when compared to `knn` or `iknn`. This suggests that each of the proposed method holds its own against most benchmark ones, especially under rank comparisons, but the cross-validation procedure adds another layer of improvement in imputation quality.

Table 2.6: Data sets considered for downstream regression and classification tasks. For classification tasks, we list the average baseline out-of-sample accuracy of an SVM model fit on the full data set, and for regression tasks, we list the average baseline out-of-sample R^2 of a LASSO model fit on the full data set.

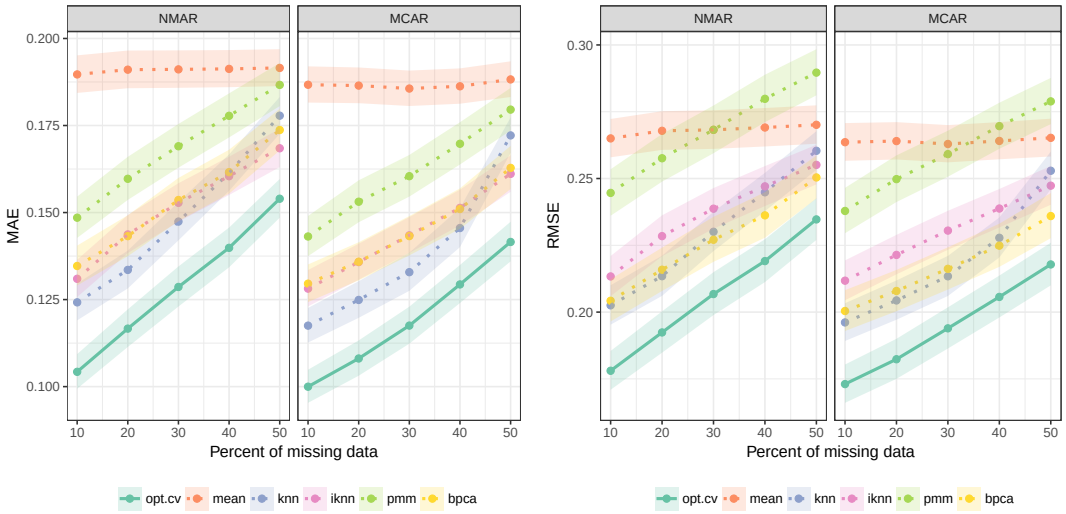
Downstream Task	Name	(n, p)	Baseline Accuracy or R^2
Classification	climate-model-crashes	(540, 18)	0.95
	connectionist-bench	(990, 10)	0.93
	ecoli	(336, 8)	0.96
	iris	(150, 4)	1.00
	pima-indians-diabetes	(768, 8)	0.77
Regression	abalone	(4177, 7)	0.51
	auto-mpg	(392, 8)	0.82
	housing	(506, 13)	0.71
	parkinsons-telemonitoring-total	(5875, 16)	0.09
	wine-quality-white	(4898, 11)	0.27

Finally, we compare against the same cross-validated procedure introduced in Section 2.2.6 applied on all the benchmark methods (`benchmark.cv`) with results in Figure 2-2b. At 30% missing data, we observe 10.1% average improvement in MAE down to 0.118 from 0.131. Further, `opt.cv` achieves highest imputation accuracy in more than 78.6% of the datasets compared to `benchmark.cv`.

Performance on Downstream Tasks

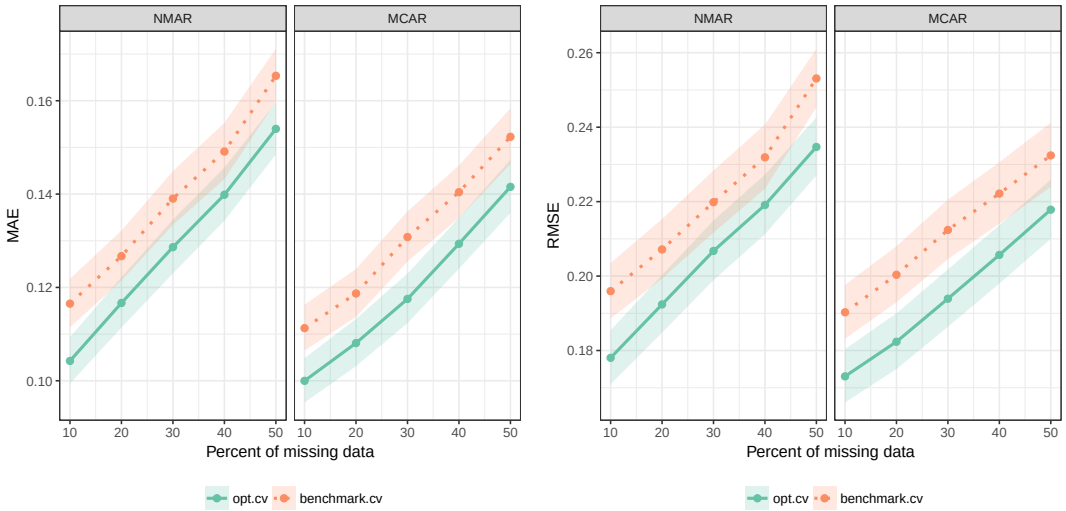
Next, we evaluate the performance of standard machine learning algorithms for classification and regression trained on the imputed data. We consider the data sets in Table 2.6, which were selected as a representative subsample from the UCI Machine Learning Repository data sets. These data sets range in size, having $n = 150$ to 5875 observations and $p = 4$ to 16 features. The difficulty of the regression or classification task on the completely known data set also varies widely. The baseline out-of-sample accuracy of an SVM model for the binary classification problems ranges from 77% to 100%, and the baseline out-of-sample R^2 of a LASSO model for the regression problems ranges from 0.09 to 0.82. For each of these data sets, the downstream tasks become more difficult as the missing data percentage increases.

In Figures 2-4 and 2-5, we show how the imputation method chosen impacts the performance for downstream tasks, across different data sets and different missing



(a) Average MAE

(b) Average RMSE



(c) Average RMSE

(d) Average RMSE

Figure 2-3: Mean absolute error (MAE) and root mean squared error (RMSE) across 95 data sets for each imputation method, comparing `opt.cv` against all benchmark methods and against the cross-validated best benchmark method, `benchmark.cv`. The center lines are geometric mean with one geometric standard deviation multiplied above and divided below. The x-axis corresponds to the percentage of missing entries.

data percentages. In Tables 2.7 and 2.8, we show pairwise t-test results, aggregating out-of-sample performance results by downstream task and missing percentage. These results include comparisons for both single and multiple imputation methods.

For the single imputation methods, we observe that `opt.cv` significantly outperforms the best cross-validated benchmark method for all missing percentages in both classification and regression tasks. Moreover, this improvement in out-of-sample accuracy and R^2 is monotonically increasing with the missing percentage. At 50% missing data, the average improvement in out-of-sample accuracy is 1.7% for classification tasks, and the average improvement in out-of-sample R^2 is 0.024 for regression tasks.

For the multiple imputation methods, we observe that `opt.mi` significantly outperforms `mice` for all missing percentages in the regression tasks, and 3/5 missing percentages in the classification tasks. At the 50% missing percentage, the average improvement is 0.5% in out-of-sample accuracy for classification tasks and 0.010 in out-of-sample R^2 for regression tasks. While these improvements are smaller than those for single imputation, they are significant at the $p = 0.001$ level.

Overall, these results suggest that `opt.impute` leads to gains in out-of-sample performance in both single and multiple imputation settings. The relative improvements are consistently greatest at the highest missing percentages, where the imputation method selected has the largest impact on the downstream performance.

Finally, we compare the performance of single vs multiple imputation for `opt.impute`. We observe that `opt.mi` significantly outperforms `opt.cv` in 8/10 scenarios, with the largest improvements occurring at the highest missing percentages. At the 50% missing percentage, the average improvement is 0.4% in out-of-sample accuracy for classification tasks and 0.017 in out-of-sample R^2 for regression tasks. These improvements are similar to the gains in performance over `mice`.

Sensitivity to Parameters

Model performance can be impacted by various parameters. For a specific data set and model, the performance can be sensitive to hyperparameters such as the number

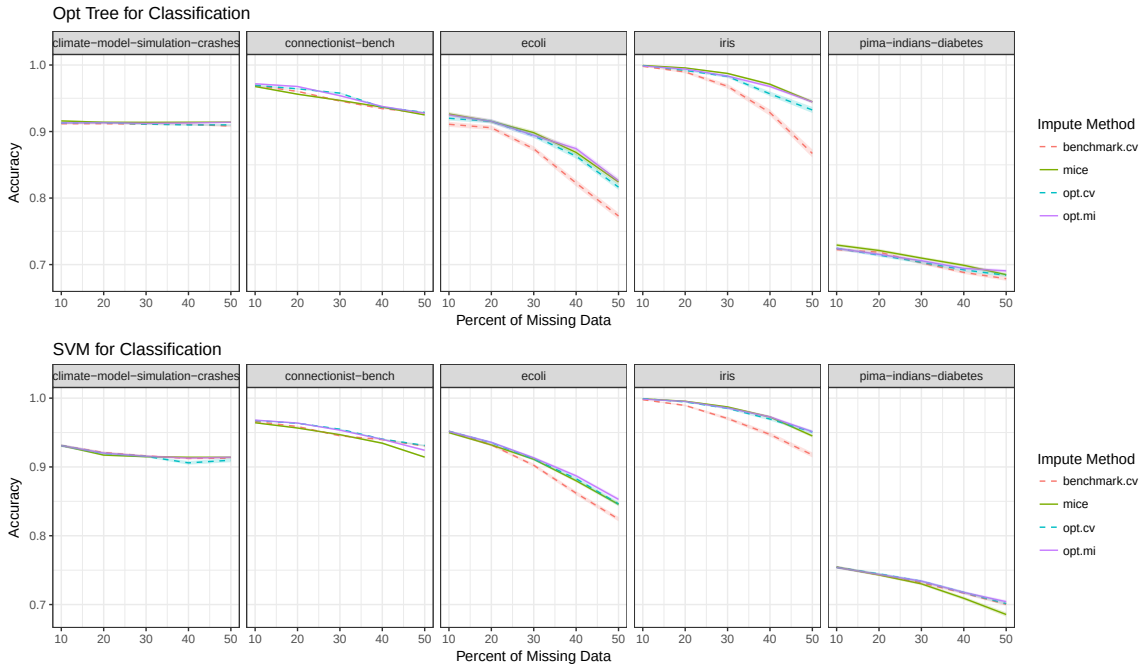


Figure 2-4: Average out-of-sample accuracy values with standard errors of SVM and Optimal Trees models trained on data imputed via `opt.impute` and benchmark methods across a sample of binary classification problems and a range of missing data percentages. Multiple and single imputation methods are solid and dotted lines respectively.

Table 2.7: Pairwise t-tests between `opt.impute` and benchmark methods for downstream classification tasks, with the p-values adjusted for multiple comparisons.

	Δ Out-of-Sample Accuracy (adjusted p-value)		
Missing %	`opt.mi - mice`	`opt.cv - benchmark.cv`	`opt.mi - opt.cv`
10	-0.0001 (1.0000)	0.0016 (0.0059**)	0.0006 (0.2076)
20	0.0018 (0.0059**)	0.0026 (<0.001***)	0.0008 (0.2076)
30	0.0005 (0.9858)	0.0082 (<0.001***)	0.0002 (1.0000)
40	0.0018 (0.0491*)	0.0113 (<0.001***)	0.0043 (<0.001***)
50	0.0052 (<0.001***)	0.0171 (<0.001***)	0.0038 (<0.001***)

Table 2.8: Pairwise t-tests between `opt.impute` and benchmark methods for downstream regression tasks, with the p-values adjusted for multiple comparisons.

	Δ Out-of-Sample R^2 (adjusted p-value)		
Missing %	`opt.mi - mice`	`opt.cv - benchmark.cv`	`opt.mi - opt.cv`
10	0.0014 (<0.001***)	0.0034 (<0.001***)	0.0013 (<0.001***)
20	0.0029 (<0.001***)	0.0113 (<0.001***)	0.0027 (<0.001***)
30	0.0071 (<0.001***)	0.0161 (<0.001***)	0.0077 (<0.001***)
40	0.0085 (<0.001***)	0.0195 (<0.001***)	0.0108 (<0.001***)
50	0.0097 (<0.001***)	0.0237 (<0.001***)	0.0174 (<0.001***)

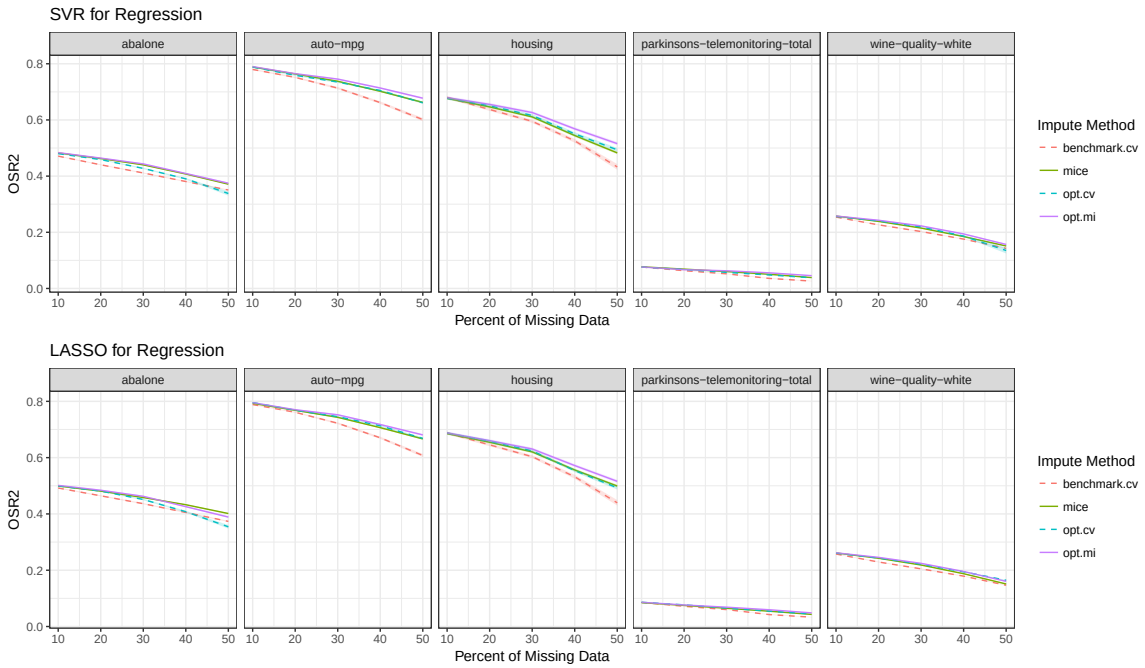
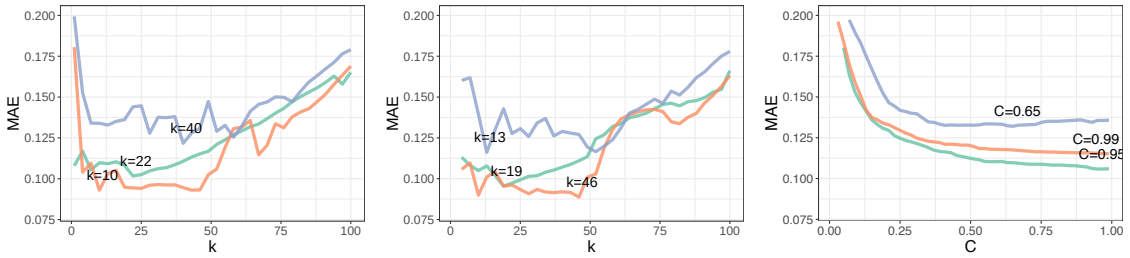


Figure 2-5: Average out-of-sample R^2 values with standard errors of SVR and LASSO models trained on data imputed via `opt.impute` and benchmark methods across a sample of regression problems and a range of missing data percentages. Multiple and single imputation methods are solid and dotted lines respectively.

of neighbors K in K-NN and the trade-off parameter C for SVM. It is also affected by the number of random starts and choice of algorithm between block coordinate descent and coordinate descent. Data characteristics such as sample size n, feature dimension p, and missing data percentage may affect the imputation quality as well. This section explores how these parameters impact the imputation quality.

We found that all of the imputation model hyperparameters that we investigated affect imputation accuracy. Figure 2-6 shows the relationship between the hyperparameters and MAE for various data sets and missing patterns. For `opt.knn` (CD and BCD), the out-of-sample MAE first decreases and then increases as the hyperparameter increases. When K reaches the sample size, the imputation is equivalent to mean imputation. For `opt.svm`, the imputation accuracy remains relatively constant with respect to changes in parameter C after a certain threshold. There were no external parameters for trees, as the trees in each step were pruned during the training process. Overall, these plots suggest that the `opt.impute` methods are relatively robust even



(a) K in `opt.knn.CD` (b) K in `opt.knn.BCD` (c) C for `opt.svm`

Figure 2-6: Sensitivity of MAE to the choice of K for the number of neighbors for K-NN coordinate descent, K-NN block coordinate descent, and the trade-off parameter C for SVM in data set `iris`. The colors represent different missing data percentages. The parameter value that achieves lowest MAE is labeled for each missing data percentage.

if their hyperparameters are not known exactly.

For `opt.knn`, the performances of block coordinate descent and coordinate descent are comparable. Under most missing data scenarios, block coordinate descent achieves the lower MAE in a few more data sets. As the missing data percentage increases, in many problems both block coordinate descent and coordinate descent methods find the same solutions, thus resulting in a tie. Comparing between the two, there is no clear dominant strategy; in practice we recommend running both methods and then selecting the imputation which yields the lowest objective value.

Computational Speed

Next, we compare the computational time required for all imputation methods across a selection of six UCI data sets and missing data patterns. Each method was run on a single thread of a machine with an Intel Xeon CPU E5-2650 (2.00 GHz) Processor and limited to 8 GB RAM with a time limit of 4 hours. For various `opt.impute` methods, we report the running times for `mean` warm starts, as multiple warm starts can be trivially parallelized. The results are shown below in Table 2.9.

Mean imputation is almost instantaneous and is therefore not presented in the table. For small-scale problems on the `iris` data set, all imputation methods finish quickly. As the data dimension p increases (for example, in the `libras-movement` data set), most `opt.impute` methods scale better than the `pmm` method. As the

Table 2.9: Computational time comparison of benchmark and `opt.impute` imputation methods. Blank entries indicate that the method failed to converge with the 4 hour time limit.

			Time (in seconds)						
			Benchmark			`opt.impute`			
Name	(n, p)	Missing %	`bpca`	`knn`	`pmm`	`knn.CD`	`knn.BCD`	`svm.CD`	`tree.CD`
iris	(150, 4)	10	0.802	0.088	0.353	0.006	0.023	0.131	0.049
		30	1.717	0.446	0.474	0.036	0.041	0.498	0.091
		50	1.875	0.736	0.334	0.085	0.097	0.762	0.062
banknote-authen.	(1372, 4)	10	2.262	2.552	1.717	0.261	1.285	3.269	0.046
		30	14.058	14.914	1.911	0.772	4.981	15.625	0.116
		50	17.820	16.889	2.141	1.578	17.573	15.280	0.159
libras-movement	(360, 90)	10	2.624	0.088	0.353	0.006	0.023	0.131	0.049
		30	3.423	0.446	0.474	0.036	0.041	0.498	0.091
		50	1.892	0.736	0.334	0.085	0.097	0.762	0.062
mushroom	(5644, 76)	10	26.432	387.386	4782.855	8.037	72.169	1442.942	-
		30	46.726	8.134	1068.476	12.818	17.572	-	-
		50	63.556	10.155	893.243	10.511	12.948	-	-
skin-segmentation	(245057, 3)	10	392.310	1144.120	12193.105	1144.144	144.679	-	9.574
		30	450.584	1380.138	-	1420.641	-	-	15.616
		50	615.037	2503.464	-	2582.102	-	-	17.818
cnae-9	(1080, 856)	10	30.310	13.038	-	12.701	12.727	-	-
		30	58.205	13.970	-	13.931	13.972	-	-
		50	126.059	14.361	-	14.284	14.343	-	-

sample size n increases, `opt.knn.CD` also scales better than `pmm`, as seen in `banknote-authentication` and `skin-segmentation`. Among the `opt.impute` methods, tree based imputation scales very well with respect to sample size n but not dimension p. Despite its high imputation quality, SVM based imputation scales relatively poorly with respect to both n and p. Among the proposed methods, `opt.knn.CD` has the best scalability in both n and p.

In particular, when comparing coordinate descent and block coordinate descent methods, the former performs best when the data size is large. When n is in the 100,000s, the coordinate descent method still converges within one hour (see `skin-segmentation`). For the block coordinate descent method, each iteration requires solving a separate system of linear equations for each continuous dimension, or an integer optimization problem for each of the categorical dimensions. On the other hand, the main bottleneck of `opt.knn.CD` is computing the K-NN assignment on $\mathbf{X}$ to update $\mathbf{Z}$ each iteration, which requires only $O(n \log n)$ time. When the problem size is small, the running times of the two methods are comparable, and the block coordinate descent method is slightly faster because it converges in fewer iterations.

However, when the number of data entries to be imputed exceeds a certain threshold, the block coordinate descent method slows down and takes much longer. In practice, we recommend running both when $n \leq 10{,}000$ and performing model selection between the two, and running only coordinate descent when n is larger.

2.4 Discussion

One of the primary contributions of this chapter is the formulation of the missing data problem as a family of optimization problems. This framework accommodates almost any predictive model that describes the conditional relationship within the data, ranging from parametric to fully non-parametric models. By design, these formulations admit arbitrary missing pattern and mixed data types and do not require specific joint distributional assumptions on the data. In addition, we show how these methods can be used to generate multiple imputations.

The first-order methods that we developed to solve these optimization problem are highly scalable and produce high quality solutions. These methods are computationally fast; for example, the coordinate descent method for SVM solves problems with 100,000s of data points and 1,000s of features in seconds on a standard desktop computer. With more random starts, we obtain solutions which continue to improve upon the objective. Since random warm starts can be trivially parallelized, increasing the number of warm starts does not change the computational times materially if implemented efficiently.

For single imputation, we propose `opt.cv`, a combination method which uses cross-validation to select the best imputation objective function from K-NN, SVM, and decision tree models. We provide evidence on `opt.cv`'s strong empirical performance against benchmark single imputation methods in large scale computational experiments on 95 real-world data sets. For all of the missing data scenarios considered, `opt.cv` produces the best overall imputation for the largest number of data sets. In addition, `opt.cv` produces the lowest average MAE and RMSE for the majority of missing data scenarios. Our proposed cross validation procedure generates additional

missing pattern under MCAR, which may be further improved by adapting the generative procedure for more accurate reflection of imputation quality in the original data missing.

Further, we demonstrate that using the imputations produced by `opt.cv` with values closer to the ground truth leads to gains in out-of-sample performance on downstream regression and classification tasks. This suggests that at medium-to-high missing percentage scenarios, machine learning practitioners will benefit significantly by adopting this framework for single imputation.

For multiple imputation, we propose `opt.mi`, a method which runs `opt.impute` on a set of probabilistically generated warm starts. We show that this method offers significant improvement over both `mice` and `opt.cv` in the downstream tasks. However, the multiple imputation methods have drawbacks because they are computationally slower, require pooling after analyzing multiple data sets, and produces an ensemble of models which is less interpretable than a single model. Therefore, unless statistical inference is required, `opt.cv` may be preferable for many applications.

Given the optimization formulations introduced in this chapter, there are multiple open questions for future research. We may consider alternate cost functions for missing data imputation that reflect out-of-sample performance better. For example, in the K-NN based model, we could add a regularizer term or use the L_1 distance or Mahalanobis distance metric instead of the squared Euclidean distance metric. The tree based imputation method invites future development in fast optimal trees for convergence and better performance. Finally, solving the global optimization problem (2.1) fast and accurately for any of the three examples of non-convex, non-linear cost functions $c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X})$ proposed in this work remains an open question.

2.5 Conclusions

In summary, we frame the classical missing data problem as a non-convex optimization problem based upon a variety of predictive models. We propose a family of new imputation methods, `opt.impute`, which finds high quality solutions to this problem

using fast first-order methods. Through extensive computational experiments on 95 data sets from the UCI Machine Learning Repository, we show that `opt.impute` yields significant gains in imputation quality over state-of-the-art imputation methods, which leads to improved out-of-sample performance on downstream tasks. This approach scales to large problem sizes, generalizes to multiple imputation, and gives significant improvement over state-of-the-art methods across a broad range of missing data scenarios.

Table 2.10 Comparison of imputation methods on data sets from the UCI Machine Learning repository with 30% missing values. The lowest mean absolute error for each data set is indicated in bold.

				Benchmark					opt.impute		
Name	n	p_0	p_1	mean	pmm	bpca	knn	iknn	knn	svm	tree
acute-inflammations-1	120	1	5	0.3701	0.3626	0.2307	0.2694	0.3598	0.2285	0.2267	**0.2185**
acute-inflammations-2	120	1	5	0.3701	0.3626	0.2307	0.2694	0.3598	0.2285	0.2267	**0.2185**
airfoil-self-noise	1503	5	0	0.2332	0.2270	0.2332	0.2018	0.2054	**0.1944**	0.1949	0.2002
airline-costs	31	9	0	0.1799	0.1566	0.1054	0.1113	0.1071	**0.0970**	0.1084	0.1037
auto-mpg	392	5	2	0.2404	0.1793	0.1547	0.1623	0.1690	0.1396	0.1362	**0.1291**
balance-scale	625	4	0	**0.3011**	0.4112	**0.3011**	0.3503	0.3113	0.3701	0.3206	0.3049
banknote-authentication	1372	4	0	0.1608	0.1596	0.1608	0.1321	0.1361	**0.1117**	0.1182	0.1243
beer-aroma	23	8	0	0.2036	0.2004	0.1772	0.1773	0.1838	0.1728	0.1638	**0.1628**
blood-transfusion	748	4	0	0.1123	0.1215	0.1123	0.0945	0.0880	0.0799	0.0824	**0.0664**
breast-cancer-diagnostic	569	30	0	0.1066	0.0431	0.0558	0.0520	0.0565	0.0486	0.0512	**0.0351**
breast-cancer-prognostic	194	31	1	0.1304	0.0727	0.0850	0.0846	0.0911	0.0794	0.0682	**0.0576**
breast-cancer	683	8	1	0.2458	0.1531	**0.1318**	0.1541	0.1788	0.1367	0.1355	0.1333
climate-model-crashes	540	18	0	**0.2505**	0.3404	**0.2505**	0.2651	0.2570	0.2750	0.2921	0.2519
communities-and-crime-2	111	101	23	0.1374	0.2191	0.1137	0.0864	0.1053	0.0845	0.0875	**0.0577**
communities-and-crime	123	99	23	0.1613	0.2901	0.1327	0.0987	0.1252	0.0973	0.0936	**0.0711**
computer-hardware	209	7	1	0.1989	0.1888	0.1989	0.1824	**0.1703**	0.1917	0.1780	0.1832
concrete-compressive	103	7	0	0.2338	0.2005	0.2057	0.2053	0.1982	0.1854	0.1868	**0.1750**
concrete-flow	103	7	0	0.2338	0.2005	0.2057	0.2053	0.1982	0.1854	0.1868	**0.1750**
concrete-slump	103	7	0	0.2338	0.2005	0.2057	0.2053	0.1982	0.1854	0.1868	**0.1750**
congressional-voting-records	232	0	16	0.4357	0.4351	0.2150	0.2504	0.4357	**0.2107**	0.2449	0.3509
connectionist-bench-sonar	208	60	0	0.1629	0.1208	0.1440	0.1088	0.1219	0.1071	0.0918	**0.0905**
connectionist-bench	990	10	0	0.1506	0.1632	0.1294	0.1049	0.1001	**0.0829**	0.1143	0.1224
construction-maintenance	33	4	0	0.3614	**0.2461**	0.3638	0.3299	0.2836	0.3283	0.3250	0.3979
contraceptive-method-choice	1473	8	1	0.2767	0.2768	0.2519	0.2634	0.2336	**0.2229**	0.2263	0.2452
dermatology	358	33	1	0.2254	0.1447	0.1484	0.1212	0.1421	**0.1082**	0.1364	0.1957
diabetes	43	2	0	0.1868	0.2768	0.1868	**0.1844**	0.2095	0.2404	0.1847	0.1950
ecoli	336	7	0	0.1215	0.1224	0.0938	0.1071	0.0908	0.0990	0.1109	**0.0904**
fertility	100	7	2	0.3526	0.3854	0.3433	0.3432	0.3476	**0.3369**	0.3450	0.3665
flags	194	22	6	0.3246	0.3146	0.3246	0.2542	0.3039	**0.2475**	0.3290	0.2603
geographic-origin	1059	68	0	0.0827	0.0764	0.0599	0.0510	0.0557	0.0477	0.0584	**0.0438**
glass-identification	214	9	0	0.1140	**0.0825**	0.0956	0.0862	0.0865	0.0851	0.0923	0.0862
haberman-survival	306	3	0	0.1701	0.2258	0.1701	0.1754	**0.1663**	0.1734	0.1727	0.1696

Table 2.10 Comparison of imputation methods on data sets from the UCI Machine Learning repository with 30% missing values. The lowest mean absolute error for each data set is indicated in bold.

				Benchmark					opt.impute		
Name	n	p_0	p_1	mean	pmm	bpca	knn	iknn	knn	svm	tree
hayes-roth	132	4	0	**0.2768**	0.3719	0.2778	0.2873	0.2779	0.2965	0.2948	0.2770
heart-disease-cleveland	297	8	5	0.3261	0.3386	0.2878	0.2945	0.3023	0.2763	**0.2738**	0.3041
hepatitis	80	4	15	0.3094	0.3019	0.3094	0.2753	0.2626	**0.2573**	0.2657	0.3480
hill-valley-noise	606	100	0	0.0998	0.0105	0.0066	0.0052	0.0283	**0.0051**	0.0781	0.0114
hill-valley	606	100	0	0.0971	0.0974	0.0055	0.0042	0.0273	0.0042	0.0783	**0.0031**
housing	506	13	0	0.1821	0.1211	0.1154	0.0985	0.1042	**0.0798**	0.1049	0.1261
hybrid-price	153	3	0	0.1538	0.1605	0.1538	0.1289	**0.1069**	0.1370	0.1202	0.1231
image-segmentation	210	19	0	0.1450	0.0806	0.0856	0.0637	0.0672	**0.0627**	0.0846	0.0628
immigrant-salaries	35	3	0	0.2247	0.2134	0.2247	0.1869	0.1700	0.1901	**0.1673**	0.1808
indian-liver-patient	579	8	2	0.1039	0.0953	0.0954	0.0981	0.0873	0.0910	0.1167	**0.0789**
ionosphere	351	34	0	0.2016	0.1739	0.1552	**0.1107**	0.1187	0.1172	0.1206	0.1475
iris	150	4	0	0.2200	0.1292	0.1571	0.1274	0.1370	0.1132	**0.1048**	0.1130
japan-emmigration	45	5	0	0.2096	0.2625	0.2098	0.2064	**0.1737**	0.2097	0.1866	0.2131
lenses	24	0	4	0.6607	0.6667	0.6696	**0.6339**	0.6607	0.6786	0.6786	0.6667
libras-movement	360	90	0	0.1823	0.0304	0.1022	0.0670	0.1014	0.0688	0.0522	**0.0139**
lpga-2008	157	6	0	0.1459	0.1769	0.1424	0.1448	0.1414	0.1496	**0.1294**	0.1299
lpga-2009	146	11	0	0.1750	0.1048	0.1074	0.1169	0.1131	0.1047	0.0889	**0.0881**
lung-cancer	27	0	56	0.3677	0.3475	0.3644	0.3426	0.3677	0.3586	**0.3348**	0.3438
mammographic-mass	830	0	5	0.2803	0.3307	0.2691	0.2386	0.2762	0.3390	0.2439	**0.2243**
monks-problems-1	124	0	6	0.6441	0.6396	0.6441	0.6059	0.6441	0.6411	**0.5991**	0.6502
monks-problems-2	169	0	6	0.6405	0.6373	0.6454	**0.6340**	0.6405	0.6481	0.6438	0.6383
monks-problems-3	122	0	6	0.6554	**0.5976**	0.6554	0.6813	0.6554	0.6577	0.6877	0.6622
parkinsons-telemonitoring-motor	587516	0		0.0623	0.0395	0.0372	0.0389	0.0342	0.0301	0.0458	**0.0265**
parkinsons-telemonitoring-total	587516	0		0.0623	0.0395	0.0372	0.0389	0.0342	0.0301	0.0458	**0.0265**
parkinsons	195	21	0	0.1348	0.0888	0.0849	0.0754	0.0814	**0.0690**	0.0824	0.0691
pima-indians-diabetes	768	8	0	0.1217	0.1453	0.1109	0.1164	0.1098	0.1089	**0.1049**	0.1069
planning-relax	182	12	0	0.1441	0.0823	0.1143	0.1188	0.1195	0.1019	0.0809	**0.0680**
post-operative-patient	87	0	8	0.3891	0.4428	0.3891	0.4143	**0.3861**	0.3937	0.4348	0.3955
pyrim	74	27	0	0.1798	0.1235	0.1758	0.1172	0.1193	**0.1145**	0.1219	0.1282
qsar-biodegradation	1055	41	0	0.0749	0.0379	0.0656	0.0385	0.0410	**0.0324**	0.0566	0.0452
seeds	210	7	0	0.2082	0.0795	0.0651	0.1099	0.0862	0.0715	0.0730	**0.0644**
soybean-large	266	0	35	0.2880	0.2583	0.2467	0.1874	0.2880	**0.1858**	0.1865	0.2103
soybean-small	47	0	35	0.2689	0.2816	0.2673	0.1577	0.2689	**0.1571**	**0.1571**	0.1837
spect-heart	80	0	22	0.2173	0.2134	0.2083	0.1899	0.2173	0.1951	**0.1869**	0.1913
spectf-heart	80	44	0	0.1307	0.1631	0.1307	0.1226	0.1195	0.1141	**0.1058**	0.1138
statlog-project-landsat-satellite	4435	36	0	0.1556	0.0405	0.0472	0.0390	0.0480	0.0329	0.0345	**0.0293**

Table 2.10 Comparison of imputation methods on data sets from the UCI Machine Learning repository with 30% missing values. The lowest mean absolute error for each data set is indicated in bold.

				Benchmark					opt.impute		
Name	n	p_0	p_1	mean	pmm	bpca	knn	iknn	knn	svm	tree
teaching-assistant-evaluation	151	1	4	0.4017	0.4074	0.4094	0.3711	0.3992	0.4086	0.5131	**0.3370**
thoracic-surgery	470	3	13	0.1469	0.1704	**0.1388**	0.1433	0.1463	0.1415	0.2205	0.1397
thyroid-disease-ann-thyroid	3772	21	0	0.0773	0.0774	0.0869	0.0723	**0.0603**	0.0838	0.1162	0.0729
thyroid-disease-new-thyroid	215	5	0	0.0935	0.1083	0.0887	0.0849	**0.0754**	0.0774	0.0893	0.0851
triazines	186	60	0	0.1574	0.0667	0.1184	0.0503	0.0708	**0.0454**	0.0892	0.0495
tv-sales	31	8	0	0.2073	0.1949	0.1808	0.1934	**0.1729**	0.1952	0.1731	0.1964
vote-for-clinton	270	49	0	0.0644	0.0715	0.0538	0.0676	0.0552	**0.0523**	0.0633	0.0537
wall-following-robot-2	5456	2	0	0.0721	0.0955	0.0721	0.0754	0.0720	0.0792	0.0847	**0.0717**
wall-following-robot-24	5456	4	0	0.0917	0.1172	0.0917	0.0886	0.0872	**0.0862**	0.0946	0.0895
wiki4he	176	0	44	0.2200	0.2234	0.1857	0.1872	0.1968	0.1777	**0.1731**	0.2085
wine-quality-red	1599	11	0	0.0976	0.0945	0.0761	0.0796	0.0744	**0.0683**	0.0757	0.0742
wine-quality-white	4898	11	0	0.0756	0.0782	0.0668	0.0771	0.0645	0.0598	0.0676	**0.0597**
wine	178	13	0	0.1680	0.1537	0.1203	0.1184	0.1144	**0.1091**	0.1105	0.1296
yacht-hydrodynamics	308	6	0	0.2102	0.1991	0.2088	0.1858	0.1861	0.1866	0.1867	**0.1799**
yeast	1484	8	0	0.0721	0.0917	0.0689	0.0740	**0.0671**	0.0683	0.0928	0.0680
zoo	101	1	15	0.2892	0.2832	0.1835	0.1518	0.2860	0.1502	0.3637	**0.1478**

Chapter 3

Robust Classification

This work, co-authored with Dimitris Bertsimas, Jack Dunn, and Colin Pawlowski, is in revisions with the INFORMS Journal on Optimization [9].

Motivated by the fact that there may be inaccuracies in features and labels of training data, we apply robust optimization techniques to study in a principled way the uncertainty in data features and labels in classification problems, and obtain robust formulations for the three most widely used classification methods: support vector machines, logistic regression, and decision trees. We show that adding robustness does not materially change the complexity of the problem, and that all robust counterparts can be solved in practical computational times. We demonstrate the advantage of these robust formulations over regularized and nominal methods in synthetic data experiments, and we show that our robust classification methods offer improved out-of-sample accuracy. Furthermore, we run large-scale computational experiments across a sample of 75 data sets from the UCI Machine Learning Repository, and show that adding robustness to any of the three non-regularized classification methods improves the accuracy in the majority of the data sets. We observe the most significant gains for robust classification methods on high-dimensional and difficult classification problems, with an average improvement in out-of-sample accuracy of robust vs. nominal problems of 5.3% for SVM, 4.0% for logistic regression, and 1.3% for decision trees.

3.1 Introduction

Three of the most widely used classification methods are SVM (Support Vector Machines), logistic regression, and CART (Classification and Regression Trees) [49]. These classifiers are among the state-of-the-art machine learning methods, giving high out-of-sample accuracy on many real-world data sets and admitting tractable training algorithms for large-scale problems. However, in many scenarios, the training data are subject to uncertainty which can negatively affect the performance of these classifiers. Regularization is a common technique for mitigating the effect of data uncertainty and addressing the problem of overfitting. In this chapter, we propose a novel approach for developing improved classifiers using techniques from robust optimization to explicitly model uncertainty in the data in a principled manner.

Support vector machines were first introduced by [36] and have gained popularity since then. SVM classifiers find a hyperplane that maximizes the margin of separation and use a hinge loss function when the data are not separable. Alternatively, the geometric concept of margin can be viewed as a form of regularization. Previous work has shown the equivalence between support vector machines and a robust formulation of the hinge loss classifier [99]. In this chapter, we develop new robust formulations for SVM and other classifiers which lead to further gains in out-of-sample accuracy compared to non-robust methods.

Logistic regression is one of the oldest and most widely used classification methods that models the probability of a response belonging to a certain class. The performance of logistic regression can be improved by introducing a regularization term to penalize model complexity, and the resulting problem can be solved efficiently for large scale problems [48]. Decision trees, a family of classification methods, aim to partition the space recursively and make predictions based on the region into which the points fall. Popular methods such as CART [29] construct the partitions with greedy heuristic methods, although recently methods have been developed that efficiently find globally optimal solutions to the decision tree problem [7]. In practice, scientists and researchers apply these methods to real-world problems using packages

which have been developed in R and other programming languages. Methods for SVM, logistic regression, and CART are included in the R packages E1071, STATS, and RPART, respectively.

The model training problems for SVM, logistic regression, and decision trees can all be formulated and solved as traditional optimization problems, and therefore can benefit from the systematic improvements in model formulation and solver speeds in this area. Recent studies have explored using modern Mixed Integer Optimization (MIO) methods to solve problems in classical statistics such as the Least Quantile Squares [15] and Best Subset Selection problems [14], and to create algorithmic approaches for fitting regression models [12, 13]. These methods have been successful in part due to dramatic increases in hardware and software computing power for MIO over the past 30 years.

One of the biggest challenges in the field of machine learning is to design models that avoid the issue of *overfitting*, where the model describes the noise instead of the underlying relationship. Strong models should take into consideration the noise structure during model estimation, and in many real-world problems, the data representing both the feature variable $(\mathbf{x}_i, i = 1, \ldots n)$ as well as the label variable $(y_i, i = 1, \ldots, n)$ are subject to error. For example, the "Wisconsin Diagnostic Breast Cancer" data set is widely used in the machine learning community. This data set involves classifying benign and malignant tumors, with features computed from digitized images including the radius, texture, symmetry, etc. of the cell nuclei. Even though the features in this data set are relatively precisely measured, the images are not free from noise, and the accuracy of the measurements depends on the precision of the recognition programs. More generally, in data sets with missing data that require imputation, uncertainties are also introduced.

As an example of label uncertainty, in the `contraceptive-method-choice` data set from the UCI machine learning repository, women were surveyed to report their current contraceptive method choice as well as demographic and socio-economic characteristics. Because of the survey nature of the data, we may suspect that some respondents have reported dishonest answers to the questions about their choice of

contraceptive method. In cancer clinical trials, caregivers determine whether or not each patient has achieved remission, and these labels are subjective and depend upon the accuracy of the tumor measurement. Another common source for such errors is the employment of labeling personnel to provide labels for the training set. Therefore, it seems natural to expect that some of the labels may be incorrect when training the classifier.

Related Work

To date, there has not been a principled way of modeling data uncertainty directly for classification problems in the literature. In this chapter, we propose a framework based on robust optimization to address classification problems whose data (both in features and in labels) are subject to error. Robust optimization is a flexible framework for modeling uncertainty [2] and is arguably one of the fastest growing areas of optimization in the last decade. For a wide variety of problems in domains such as finance, statistics, and health care, robust formulations have been shown to be computationally tractable and lead to improved solutions compared to the classical optimization formulations [5]. The key advantage of robust solutions is that they provide near optimal solutions that remain feasible when problem parameters are perturbed, and thus are attractive when the problem is subject to uncertainty.

In particular, robust optimization has been shown to lead to improvements for many statistics problems. In the machine learning community, the success of SVM in classification and Lasso in regression has been largely attributed to their regularization terms that reduce data overfitting. [77] demonstrate how robust classification can be used to handle situations with imbalanced training data, and [70] derive classifiers protected against stochastic adversarial perturbations to the training data. [99] establish that robustness is a key *reason* behind the strong performance of regularized methods, due to the generalization ability of robustness.

There has been prior work which consider robust optimization classifiers based upon SVM, first proposed in [24, 23]. These approaches have dealt mainly with feature uncertainty. One of the robust classification methods proposed in this work,

namely feature-robust SVM, closely resembles the linear optimization robust classifiers proposed by [91], except these methods contain an additional regularizer term in the objective. This difference is important because more recently, it has been shown that a robust optimization formulation of the maximum margin classifier is equivalent to the classical SVM; thus methods derived as robust variations to classical SVM are "double-counting" the effect of robustness [99, 6]. In addition, there have been previous attempts to model uncertainties in labels for SVM, although these methods are largely heuristic in nature and have been tested primarily on synthetic or contaminated data [25, 74]. There has also been work on robustifying kernel SVM methods against feature uncertainty by [1]. The approach we present could be extended to kernel methods, but this is beyond the scope of the work.

For logistic regression, regularized versions such as Elastic Net have been proposed [103], which consider adding a convex combination of the ℓ_1 and ℓ_2-norm penalty to the objective; however these regularized classifiers were not derived using tools from robust optimization. Using robust optimization, logistic regression models that are robust to feature uncertainty have been derived for various uncertainty sets [42, 54].

To our knowledge, no work has been done framing decision trees as a robust optimization problem. Because tractable formulations and solution methods for the optimal decision tree problem were proposed quite recently in [7], robust optimal decision trees have not been explored.

In summary, results from the literature indicate that ideas from robust optimization have the potential to add value to existing classification methods. Prior work on SVM establishes the equivalence between regularization and robustness for certain problems, and in some examples robust classifiers yield higher out-of-sample accuracy compared to nominal methods. However, these works have largely focused on theoretical derivations of robust methods, with limited testing on synthetic data. Without extensive computational experiments, we do not know if these robust classifiers yield gains in out-of-sample accuracy in practice, especially in comparison with regularized methods.

We build upon these previous efforts to present a framework for robust classi-

fication which accommodates three of the most widely used classification methods: SVM, logistic regression, and CART. By considering a diverse variety of classifiers, we compare the impact of adding robustness to different models, and we evaluate the performance of these methods in practice through large-scale computational experiments.

Contributions

This chapter shows how to incorporate robustness in classification problems generally. Under the framework of robust optimization, we systematically develop new robust methods that offer predictable improvements in out-of-sample accuracy over nominal classifiers. We summarize our contributions below:

1. We present a principled framework for robust classification, which combines ideas from robust optimization and machine learning, with an aim to build classifiers that model data uncertainty directly. Building on previous work for modeling feature uncertainty, we introduce an approach for modeling uncertainty in labels, as well as both features and labels simultaneously. By viewing machine learning algorithms as a family of optimization problems, we show that the robustification of existing classification methods can be done in a unified and principled way. This leads to tractable problems with relatively small overhead compared to the original methods. In particular, we use this framework to derive counterparts to SVM, logistic regression, and CART that are robust to variations in features and labels in the data. In the case where we consider feature uncertainty only, the resulting robust formulations for SVM and logistic regression match previous results in the literature.

2. We demonstrate the advantage of robust formulations over regularized and nominal methods through synthetic data experiments with two classes divided by a separating hyperplane. Compared to nominal and regularized methods, the robust SVM and logistic regression methods recover the separating hyperplane classifiers closer to the truth, leading to gains in out-of-sample accuracy especially in the worst case analysis.

3. We demonstrate that robust classification improves out-of-sample accuracy in large-scale computational experiments across a sample of 75 data sets from the UCI Machine Learning Repository. Furthermore, we identify characteristics of classification problems for which robust methods lead to significant accuracy gains compared to non-robust methods. Specifically, in problems with high dimensional data and difficult separability, the value of robustness is even more prominent.

4. We provide a simple, empirically-derived decision rule for machine learning practitioners that predicts with high accuracy when robust methods can offer significant improvement over the nominal methods, with an average improvement in out-of-sample accuracy of 5.3% for SVM, 4.0% for logistic regression, and 1.3% for CART. Compared to regularized SVM or logistic regression, the average out-of-sample accuracy improvement of our principled approach to robustness is 2.1% over regularized SVM and 1.2% over regularized logistic regression when this rule is satisfied.

We would like to distinguish robust optimization in statistical problems from the field of robust statistics, developed by [57], which studies how an estimator performs under perturbation of the model. Even though both fields share the motivation to avoid unduly effects from outliers, the underlying methodologies are totally different and address the problems from separate angles. While robust statistics passively evaluates the robustness properties of a given algorithm, robust optimization actively constructs models which take into account data uncertainty.

The structure of the chapter is as follows. In Section 3.2, we present a selection of widely-used classification methods. In Section 3.3, we give a brief introduction to robust optimization and introduce some terms and properties that will be used later. In Section 3.4, we demonstrate how to apply robust optimization to the classification methods to derive a family of classification methods that are robust to uncertainty in the features of the training data set. In Section 3.5, we repeat this process to develop methods that are robust to uncertainty in data set labels. In Section 3.6, we combine

these approaches to develop classification methods that are robust to noise in both features and labels. In Section 3.7, we compare the performance of these robust classification methods to their non-robust counterparts and regularized methods through a series of synthetic data experiments. In Section 3.8, we comprehensively compare the performance of our robust classifiers to their benchmark methods on a wide range of real data sets. We conclude in Section 3.9.

3.2 Overview of Classification Methods

In this section, we present a selection of widely-used methods for classification. These are the methods to which we will later apply robust optimization techniques. For this section and in the rest of the chapter, let $\{\mathbf{x}_i, y_i\}_{i=1}^n$ be the training data provided for the classification task, where $\mathbf{x}_i \in \mathbb{R}^p$ is the feature vector and $y_i \in \{-1, 1\}$ is the label for observation i.

3.2.1 Soft-Margin Support Vector Machines

Soft-margin support vector machines are a variation on the simpler maximal margin classifier which relax the requirement that the data be separable and instead allow for points to be incorrectly classified [36]. Support vector machines use hinge loss as the loss function, and balance the minimization of total loss and maximization of margin with parameter C that can be tuned via validation. This classifier can be formulated as the following problem:

$$\min_{\mathbf{w},b} \quad \tfrac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \max\{1 - y_i(\mathbf{w}^T\mathbf{x}_i - b), 0\}. \tag{3.1}$$

Problem (3.1) can equivalently be formulated as the following problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \tfrac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i \quad i = 1,\ldots,n, \\ & \xi_i \geq 0 \qquad\qquad\qquad\quad i = 1,\ldots,n. \end{aligned} \tag{3.2}$$

The dual problem can be formulated through the use of Lagrange multipliers:

$$\begin{aligned} \max_{\alpha} \quad & C\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \quad i = 1,\ldots,n, \\ & \sum_{i=1}^{n}\alpha_i y_i = 0. \end{aligned}$$

Both the primal and dual are convex quadratic optimization problems. Since the dual problem has fewer decision variables, and the majority of these variables tend to be equal to zero or the cost parameter C in the optimal solution, it is typically the problem solved in practice [49]. In addition, the dual form is advantageous because it allows us to do the kernel trick to learn non-linear decision rules [36]. Alternatively, we may modify the objective function of problem (3.1) by changing the norm of the regularizer term from ℓ_2 to ℓ_1 [102]. The resulting classifier is formulated as follows:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|_1 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i \quad i = 1,\ldots,n, \\ & \xi_i \geq 0 \qquad\qquad\qquad\quad i = 1,\ldots,n. \end{aligned} \tag{3.3}$$

Problem (3.3), which we refer to as ℓ_1-regularized SVM, is equivalent to a linear optimization problem which is efficiently solvable.

3.2.2 Logistic Regression

Logistic regression assumes the response variable Y follows a Bernoulli distribution with the probability depending on the $\mathbf{x}$ and the model parameter $\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}$

$$\begin{aligned}\mathbb{P}(Y = 1|\mathbf{X} = \mathbf{x}) &= \frac{e^{\beta^T\mathbf{x}+\beta_0}}{1 + e^{\beta^T\mathbf{x}+\beta_0}},\\ \mathbb{P}(Y = -1|\mathbf{X} = \mathbf{x}) &= \frac{1}{1 + e^{\beta^T\mathbf{x}+\beta_0}}.\end{aligned}$$

Concisely, the conditional probability can be written as

$$\mathbb{P}(Y = y_i|\mathbf{X} = \mathbf{x}) = \frac{1}{1 + e^{-y_i(\beta^T\mathbf{x}_i+\beta_0)}}.$$

Logistic regression coefficients β and β_0 are typically fit using maximum likelihood method. The log-likelihood is

$$-\sum_{i=1}^{n} \log\left(1 + e^{-y_i(\beta^T\mathbf{x}_i + \beta_0)}\right).$$

Therefore, the maximum-likelihood estimators β and β_0 aim to solve the following problem:

$$\max_{\beta,\beta_0} \; -\sum_{i=1}^{n} \log\left(1 + e^{-y_i(\beta^T\mathbf{x}_i + \beta_0)}\right). \tag{3.4}$$

Problem (3.4) is a concave maximization problem that is efficiently solvable by methods such as coordinate descent or Newton's method [4].

Similar to the regularization techniques in the popular lasso regression [90] for variable selection and shrinkage, a regularization term can be added to the logistic regression likelihood function, giving

$$\max_{\beta,\beta_0} \; -\sum_{i=1}^{n} \log\left(1 + e^{-y_i(\beta^T\mathbf{x}_i + \beta_0)}\right) - \lambda\|\beta\|_q, \tag{3.5}$$

where $\|\cdot\|_q$ is a given ℓ_q norm.

3.2.3 Decision Trees and CART

Decision Trees are a family of classification methods that seek to recursively partition the feature space into disjoint regions and predict labels for new points based upon the region into which the point falls. The most widely-used method for training decision trees is CART [29], which takes a greedy heuristic approach to constructing the tree rather than posing the entire process as a single optimization problem.

However, in order to use robust optimization techniques to create robust decision trees, we require the formulation of the decision tree training problem as a formal optimization problem. Optimal Decision Trees [7] are a recent method that considers the entire decision tree learning procedure as a single mixed-integer optimization problem, and uses this to take a globally optimal view while constructing the tree. To create robust decision tree methods, we will take the Optimal Decision Tree problem and apply robust optimization.

Consider the problem of training a general decision tree. At each branch node in the tree, a split of the form $\mathbf{a}^T\mathbf{x} < b$ is applied. Points that satisfy this constraint will follow the left branch of the tree, while those that violate the constraint follow the right branch. Each leaf node is assigned a label, and each point is assigned the label of the leaf node into which the point falls. Figure 3-1 summarizes this for an example decision tree with two branch nodes, A and B, that apply splits $\mathbf{a}_A^T\mathbf{x} < b_A$ and $\mathbf{a}_B^T\mathbf{x} < b_B$ respectively. There are three leaf nodes that assign labels {-1}, {+1}, and {+1} (from left to right in the figure).

Given that the tree contains K nodes, we define the sets $\mathcal{P}_k^L$, $\mathcal{P}_k^R$, and $\mathcal{P}_k$ for $k = 1, \ldots, K$ to capture the hierarchy of the tree

- $\mathcal{P}_k^L$ = the ancestors of node k in the tree of which we have taken the left branch (a split of the form $\mathbf{a}_k^T\mathbf{x}_i < b_k$) to get to node k;
- $\mathcal{P}_k^R$ = the ancestors of node k of which we have taken the right branch (a split of the form $\mathbf{a}_k^T\mathbf{x}_i \geq b_k$) to get to node k;
- $\mathcal{P}_k = \mathcal{P}_k^L \cup \mathcal{P}_k^R$, i.e., all ancestors of node k.

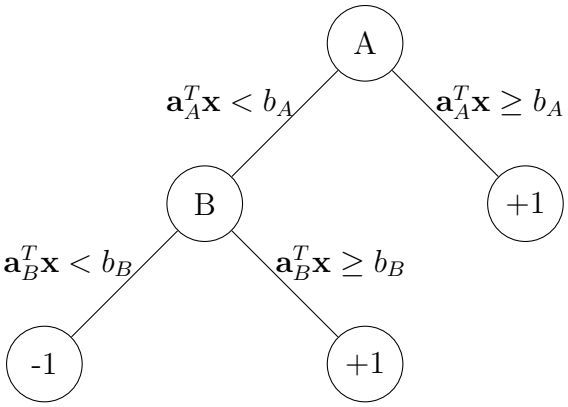


Figure 3-1: An example of a decision tree with two partition nodes and three leaf nodes.

We will now state the Optimal Decision Tree problem from [7] below as Problem (3.6) and then provide an explanation of the model:

$$\min \quad \sum_{k=1}^{K} f_k - \sum_{k=1}^{K} \lambda_k d_k \tag{3.6a}$$

$$\text{s.t.} \quad g_k = \sum_{i=1}^{n} \frac{1-y_i}{2} z_{ik} \qquad k = 1, \ldots, K, \tag{3.6b}$$

$$h_k = \sum_{i=1}^{n} \frac{1+y_i}{2} z_{ik} \qquad k = 1, \ldots, K, \tag{3.6c}$$

$$f_k \leq g_k + M[w_k + (1 - c_k)] \qquad k = 1, \ldots, K, \tag{3.6d}$$

$$f_k \leq h_k + M[(1 - w_k) + (1 - c_k)] \qquad k = 1, \ldots, K, \tag{3.6e}$$

$$f_k \geq g_k - M[(1 - w_k) + (1 - c_k)] \qquad k = 1, \ldots, K, \tag{3.6f}$$

$$f_k \geq h_k - M[w_k + (1 - c_k)] \qquad k = 1, \ldots, K, \tag{3.6g}$$

$$d_k = 1 \qquad k = \lceil K/2 \rceil, \ldots, K, \tag{3.6h}$$

$$d_k \leq d_j \qquad k = 1, \ldots, K, \forall j \in \mathcal{P}_k, \tag{3.6i}$$

$$d_k + \sum_{l=1}^{p} a_{kl} = 1 \qquad k = 1, \ldots, K, \tag{3.6j}$$

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1, \ldots, n, \tag{3.6k}$$

$$z_{ik} \leq d_k \qquad i = 1, \ldots, n, k = 1, \ldots, K, \tag{3.6l}$$

$$z_{ik} \leq 1 - d_j \qquad i = 1, \ldots, n, k = 1, \ldots, K, \forall j \in \mathcal{P}_k, \quad (3.6m)$$

$$\sum_{i=1}^{n} z_{ik} \geq N c_k \qquad k = 1, \ldots, K, \quad (3.6n)$$

$$c_k \geq d_k - \sum_{j \in \mathcal{P}_k} d_j \qquad k = 1, \ldots, K, \quad (3.6o)$$

$$\mathbf{a}_j^T \mathbf{x}_i + \epsilon \leq b_j + M(1 - z_{ik}) \qquad i = 1, \ldots, n, k = 1, \ldots, K, \forall j \in \mathcal{P}_k^l, \quad (3.6p)$$

$$\mathbf{a}_j^T \mathbf{x}_i \geq b_j - M(1 - z_{ik}) \qquad i = 1, \ldots, n, k = 1, \ldots, K, \forall j \in \mathcal{P}_k^u, \quad (3.6q)$$

$$\mathbf{a}_k \in \{0, 1\}^p \qquad k = 1, \ldots, K, \quad (3.6r)$$

$$0 \leq b_k \leq 1 \qquad k = 1, \ldots, K, \quad (3.6s)$$

$$z_{ik}, w_k, c_k, d_k \in \{0, 1\} \qquad i = 1, \ldots, n, k = 1, \ldots, K. \quad (3.6t)$$

At each node $k = 1, \ldots, K$ in the tree, we must decide whether to apply a split or set the node to be a leaf node. The binary variable d_k takes value 1 if no split is applied, and 0 otherwise.

If we choose to apply a split at a node k, the variables $\mathbf{a}_k$ and b_k are used to set a split of the form $\mathbf{a}_k^T \mathbf{x} < b_k$. To mirror the behavior of CART, we only consider univariate decision trees and hence we only allow a single variable to be used in each split. This is achieved by the constraints (3.6r), which forces the components of $\mathbf{a}_k$ to be binary, and (3.6j) means we can only choose one of these variables at each node. Note that (3.6j) also forces $\mathbf{a} = \mathbf{0}$ if $d_k = 1$, so we cannot apply a split at a node that has been marked as a leaf node.

We use the binary variables z_{ik} to track which leaf node k each point $i = 1, \ldots, n$ in training set is assigned. Constraints (3.6p) and (3.6q) ensure that points are assigned only to a node if they satisfy all required splits, while constraints (3.6l) and (3.6m) ensure that points can only be assigned to leaf nodes. Finally, (3.6k) ensures that each point is assigned to exactly one leaf node.

The objective is to minimize the number of misclassified points. The number of misclassified points in a node k is tracked using the variable f_k. Note that it is always better to assign the leaf node a label that agrees with the most common label among points in the node. This means the misclassification count is given by the size of

the minority label. We use the variables g_k and h_k to count the number of points of each label in each node k, which is achieved with constraints (3.6b) and (3.6c). Constraints (3.6d) through (3.6g) set f_k to $\min\{g_k, h_k\}$ to count the misclassification in each node, and the objective sums this misclassification over all nodes.

CART imposes a constraint relating to the `minbucket` parameter, which requires each leaf node to contain at least this number of points. Constraints (3.6n) and (3.6o) enforce this restriction in the model for a supplied `minbucket` parameter N.

The small number of remaining constraints relate to ensuring the decision to split or not at each node is permitted by the structure of the tree. For example, no leaf node is permitted to have a child node. We omit the full details of these precedence constraints from this description of the model and instead refer the reader to [7] for the complete description.

This is a mixed-integer optimization problem that is practically solvable on real-world data sets and leads to results that are highly competitive with heuristic decision tree methods like CART (see [7] for a comprehensive comparison).

3.3 Brief Overview of Robust Optimization

In this section, we give an overview of robust optimization and introduce the notions of uncertainty sets and dual norms that will be used later when applying robust optimization techniques to the unified classification framework.

Robust optimization is a means for modeling uncertainty in optimization problems without the use of probability distributions. Under this modeling framework, we construct deterministic *uncertainty sets* that contain possible values of uncertain parameters. We then seek a solution that is optimal for all such realizations of this uncertainty. Consider the general optimization problem:

$$\begin{aligned} \max_{\mathbf{x}\in\mathcal{X}} \quad & c(\mathbf{u}, \mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{u}, \mathbf{x}) \leq \mathbf{0}, \end{aligned}$$

where $\mathbf{x}$ is the vector of decision variables, $\mathbf{u}$ is a vector of given parameters, c is a real-valued function, $\mathbf{g}$ is a vector-valued function, and $\mathbf{0}$ is the vector of all zeros. Relaxing the assumption that $\mathbf{u}$ is fixed, we assume instead that the realized values of $\mathbf{u}$ are restricted to be within some uncertainty set $\mathcal{U}$. We form the corresponding robust optimization problem by optimizing against the worst-case realization of the uncertain parameters across the entire uncertainty set:

$$\begin{aligned} \max_{\mathbf{x}\in\mathcal{X}} \quad & \min_{\mathbf{u}\in\mathcal{U}} \; c(\mathbf{u},\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{u},\mathbf{x}) \leq \mathbf{0} \quad \forall \mathbf{u} \in \mathcal{U}. \end{aligned}$$

Despite typically having an infinite number of constraints, it is often possible to reformulate the problem as a deterministic optimization problem with finite size, depending on the choice of uncertainty set $\mathcal{U}$. The resulting deterministic problem is deemed the *robust counterpart*, which may be a problem of the same complexity as the nominal problem, depending on the structure of $\mathcal{U}$.

There is extensive evidence in the literature that robust solutions have significant advantages relative to nominal solutions. A case study of linear optimization problems from the NETLIB library found that in 13 out of 90 problems, the optimal non-robust solution violates some of the inequality constraints by more than 50% of the right-hand side values, when the uncertain coefficients are subject to small (0.01%) perturbations. On the other hand, robust solutions for these identical problems which are feasible for all perturbations up to 0.1% lead to objective values that are within 1% of the optimal [3].

Dual Norms

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a vector in $\mathbb{R}^n$. For any real number $q \geq 1$, we define the ℓ_q norm of $\mathbf{x}$ in the standard way, denoted by $\|\mathbf{x}\|_q$:

$$\|\mathbf{x}\|_q \triangleq \left(\sum_{i=1}^{n} |x_i|^q \right)^{\frac{1}{q}}.$$

A particular problem that is encountered frequently when using robust optimization is the so-called *dual norm* problem:

$$\max_{\|\mathbf{x}\|_q \leq 1} \{\mathbf{a}^T \mathbf{x}\}.$$

When $q > 1$, the optimal solution to this problem is $\|\mathbf{a}\|_{q*}$, where $q^* = \frac{1}{1-\frac{1}{q}}$. This ℓ_{q^*} norm is called the *dual norm* of the ℓ_q norm. In addition, when $q = 1$, it can be shown that the optimal solution to this problem is $\|\mathbf{a}\|_\infty$, where the ℓ_∞ norm of a vector $\mathbf{x} \in \mathbb{R}^n$ is defined by

$$\|\mathbf{x}\|_\infty \triangleq \lim_{q \to \infty} \|\mathbf{x}\|_q = \max\{|x_1|, |x_2|, \ldots, |x_n|\}.$$

A simple extension to this problem is when the norm of $\mathbf{x}$ is restricted by any number $\rho > 0$. In this case we have the following:

$$\max_{\|\mathbf{x}\|_q \leq \rho} \{\mathbf{a}^T \mathbf{x}\} = \max_{\|\mathbf{y}\|_q \leq 1} \{\mathbf{a}^T (\rho \mathbf{y})\} = \rho \cdot \max_{\|\mathbf{y}\|_q \leq 1} \{\mathbf{a}^T \mathbf{y}\}, \tag{3.7}$$

and the optimal solution to this problem is thus $\rho \|\mathbf{a}\|_{q^*}$.

3.4 Robustness Against Uncertainty in Features

In this section, we present the notion of robustifying classification methods against uncertainties in the features of the training set. Using an uncertainty set to model possible values of the features in reality, we then define and state the *feature-robust counterpart* for each of the classification methods. We note that the feature-robust counterparts for SVM and logistic regression are known in the literature, but we include their derivation here for completeness.

3.4.1 Motivating Feature-Robustness

Uncertainties in the features can arise from measurement errors during data collection and from input errors during data manipulation and missing value imputation. If left

unaddressed, the trained model may be biased and severely influenced by inaccuracies in the data. Our goal is to train a *feature-robust* model that takes such uncertainties into account, which is stable and provides high accuracy in circumstances where data are perturbed.

With the robust approach, such uncertainties are taken into consideration when training the classifiers. To model uncertainty in the features of the training set, we assume that the data $\mathbf{x}_i$ are subject to additive perturbations $\mathbf{\Delta x}_i \in \mathbb{R}^p,\ i = 1, \ldots, n$. Let $\mathbf{\Delta X} = (\mathbf{\Delta x}_1, \mathbf{\Delta x}_2, \ldots \mathbf{\Delta x}_n)$ and define the following uncertainty set:

$$\mathcal{U}_x = \{\mathbf{\Delta X} \in \mathbb{R}^{n \times p} \mid \|\mathbf{\Delta x}_i\|_q \leq \rho, i = 1, \ldots, n\}, \tag{3.8}$$

where ρ is a parameter controlling the magnitude of the considered perturbations, and hence the degree to which the features in the training set are able to deviate from their nominal values.

After introducing these perturbations, the features in the training set take values $\mathbf{x}_i + \mathbf{\Delta x}_i,\ i = 1, \ldots, n$. We now seek to construct a classifier that is robust to all such perturbations $\mathbf{\Delta X} \in \mathcal{U}_x$. To do this, we robustify against this uncertainty set of feature parameters in each of our classification methods. In practice, the parameter ρ can be chosen via validation, and the range to be searched over can be fixed if each feature in the data set is normalized. We also note that when $\rho = 0$, the problem is equivalent to the nominal problem, and so the nominal solution is a possible candidate to be considered during validation. This means the feature-robust classifier will only be preferred over the nominal method when the validation score is better.

In addition, note that $\mathcal{U}_x$ is the cartesian product of the sets $\{\mathbf{\Delta x}_i \in \mathbb{R}^p | \|\mathbf{\Delta x}_i\|_q \leq \rho\}, i = 1, \ldots, n$. This structure enables us to derive tractable robust counterparts for all three classification methods. We may consider alternative uncertainty sets for the feature perturbations as well, for example polyhedral or ellipsoidal uncertainty sets. Here, we consider the norm uncertainty set $\mathcal{U}_x$ because it admits a simple geometric interpretation and only requires tuning a single parameter ρ, which makes it tractable to evaluate in the computational experiments and to use in practice.

We present the reformulated robust counterparts below for soft-margin support vector machines, logistic regression, and optimal decision trees. For each method, we refer to the resulting deterministic optimization problem as the *feature-robust counterpart* of that classifier.

3.4.2 Soft-Margin Support Vector Machines

The regularized Support Vector Machine problem in (3.2) has been shown by [99] and [45] to be equivalent to the robust counterpart of a nominal problem under a particular choice of uncertainty set in the features. These results suggest that the regularization term $\|\mathbf{w}\|_2^2$ is a by-product of feature robustness. Further discussion of the equivalence between classical SVM and feature-robust formulations is provided in Section 3.10. In the following sections, to avoid double counting the effect of robustness, we consider the hinge loss classifier without the regularization term to be the nominal method for SVM:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \max\{1 - y_i(\mathbf{w}^T\mathbf{x}_i - b), 0\}. \tag{3.9}$$

Robustifying Problem (3.9) against the uncertainty set $\mathcal{U}_x$ gives the following robust optimization problem:

$$\min_{\mathbf{w},b} \max_{\Delta\mathbf{X}\in\mathcal{U}_x} \sum_{i=1}^{n} \max\{1 - y_i(\mathbf{w}^T(\mathbf{x}_i + \mathbf{\Delta}\mathbf{x}_i) - b), 0\}. \tag{3.10}$$

We now derive the robust counterpart to Problem (3.10). Note that this is equivalent to Theorem 3 in [99].

Theorem 1. *The robust counterpart to Problem* (3.10) *is*

$$\begin{aligned} \min_{\mathbf{w},b} \ & \sum_{i=1}^{n} \xi_i \\ \text{s.t. } & y_i(\mathbf{w}^T\mathbf{x}_i - b) - \rho\|\mathbf{w}\|_{q^*} \geq 1 - \xi_i \qquad i = 1, \ldots, n, \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\qquad\quad i = 1, \ldots, n. \end{aligned} \tag{3.11}$$

where ℓ_{q^*} is the dual norm of ℓ_q.

Proof. Proof. We can reformulate Problem (3.10) as

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^T(\mathbf{x}_i + \boldsymbol{\Delta}\mathbf{x}_i) - b) \geq 1 - \xi_i \quad \forall \boldsymbol{\Delta}\mathbf{X} \in \mathcal{U}_x \qquad i = 1, \ldots, n, \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad i = 1, \ldots, n. \end{aligned}$$

The first constraint must be satisfied for all $\boldsymbol{\Delta}\mathbf{X} \in \mathcal{U}_x$, thus the constraint is equivalent to

$$\min_{\boldsymbol{\Delta}\mathbf{X} \in \mathcal{U}_x} (y_i \mathbf{w}^T \boldsymbol{\Delta}\mathbf{x}_i) \geq 1 - \xi_i - y_i(\mathbf{w}^T\mathbf{x}_i - b) \quad i = 1, \ldots, n.$$

Here, for all $i = 1, \ldots, n$, the minimization term is equal to the objective value of the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\Delta}\mathbf{x}_i} \quad & y_i \mathbf{w}^T \boldsymbol{\Delta}\mathbf{x}_i \\ \text{s.t.} \quad & \|\boldsymbol{\Delta}\mathbf{x}_i\|_q \leq \rho. \end{aligned}$$

Because y_i is constant, we recognize this optimization problem as the dual norm problem. Therefore, by (3.7), for any given value of $\mathbf{w}$, the objective value of this problem is $-\rho\|\mathbf{w}\|_{q^*}$, where ℓ_{q^*} is the dual norm of ℓ_q. Replacing the minimization term with this optimal value and rearranging yields (3.11). □

Depending on the choice of norm, the feature-robust counterpart of SVM can be solved efficiently using various optimization methods. For example, when $q = q^* = 2$, feature-robust SVM can be solved using second-order cone optimization methods [4]. When $q = 1,\ q^* = \infty$ or $q = \infty,\ q^* = 1$, feature-robust SVM can be reformulated as a linear optimization problem.

3.4.3 Logistic Regression

Robustifying Problem (3.4) against the uncertainty set $\mathcal{U}_x$ yields the following robust optimization problem:

$$\max_{\beta,\beta_0} \min_{\mathbf{\Delta X}\in\mathcal{U}_x} -\sum_{i=1}^{n} \log\left(1+e^{-y_i(\beta^T(\mathbf{x}_i+\mathbf{\Delta x}_i)+\beta_0)}\right). \tag{3.12}$$

Next we determine the robust counterpart to Problem (3.12). We note that similar results on more specific uncertainty sets have been previously shown in [42, 54].

Theorem 2. *The robust counterpart to Problem* (3.12) *is*

$$\max_{\beta,\beta_0} -\sum_{i=1}^{n} \log\left(1+e^{-y_i(\beta^T\mathbf{x}_i+\beta_0)+\rho\|\beta\|_{q^*}}\right), \tag{3.13}$$

where ℓ_{q^*} is the dual norm of ℓ_q.

Proof. Proof. Consider the inner minimization problem in (3.12), which is the following optimization problem:

$$\min_{\mathbf{\Delta X}\in\mathcal{U}_x} -\sum_{i=1}^{n} \log\left(1+e^{-y_i(\beta^T(\mathbf{x}_i+\mathbf{\Delta x}_i)+\beta_0)}\right). \tag{3.14}$$

Let $\omega_i = y_i(\beta^T(\mathbf{x}_i+\mathbf{\Delta x}_i)+\beta_0)$, and define $g(\omega_i) = -\log(1+e^{-\omega_i})$. The first-order derivative of g with respect to ω_i is

$$\frac{dg}{d\omega_i} = \frac{1}{1+e^{\omega_i}},$$

which is strictly positive. Therefore, for each $i=1,\ldots,n$, the solution to the inner minimization problem in (3.12) is the same as the solution of the problem

$$\begin{aligned} \min_{\mathbf{\Delta x}_i} \quad & y_i(\beta^T(\mathbf{x}_i+\mathbf{\Delta x}_i)+\beta_0) \\ \text{s.t.} \quad & \|\mathbf{\Delta x}_i\|_q \le \rho. \end{aligned} \tag{3.15}$$

This is equivalent to the following problem:

$$y_i(\beta^T \mathbf{x}_i + \beta_0) - \max_{\mathbf{\Delta x}_i} \; - y_i \beta^T \mathbf{\Delta x}_i$$
$$\text{s.t. } \|\mathbf{\Delta x}_i\|_q \leq \rho.$$

We recognize this maximization term as the dual norm problem. Therefore, by (3.7), the optimal solution is $\rho\|\beta\|_{q^*}$, where ℓ_{q^*} is the dual norm of ℓ_q. We conclude that the optimal value to (3.15) is $y_i(\beta^T \mathbf{x}_i + \beta_0) - \rho\|\beta\|_{q^*}$. Substituting the optimal value into the inner minimization problem in (3.12), we obtain

$$-\sum_{i=1}^{n} \log\left(1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0) + \rho\|\beta\|_{q^*}}\right).$$

Maximizing the above equation over β, β_0 yields (3.13). □

If $q \geq 2$, the robust counterpart (3.13) is differentiable (as in the nominal problem) and thus is still solvable using gradient and Newton methods. However, if $q \in \{1, \infty\}$ then Problem (3.13) becomes non-differentiable and we may solve it using subgradient methods. Alternatively, we may remodel the nonlinear terms to obtain a differentiable formulation with linear constraints, which is solvable using gradient and Newton methods for constrained optimization [4].

Compared to the nominal case, the feature-robust counterpart of logistic regression has an additional $\rho\|\beta\|_{q^*}$ term in the exponent of the logit function. It resembles the regularization term in regularized logistic regression, shown in Equation (3.5). However, the additional term from robustness penalizes model complexity in the logit, or log odds ratio, while the regularization term is a linear penalty on the entire likelihood. The connection between the two can be shown via a first-order Taylor series expansion of the objective function of the feature-robust counterpart, which gives the following:

$$-\sum_{i=1}^{n} \log\left(1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}\right) - \sum_{i=1}^{n} \frac{e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}}{1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}} \rho\|\beta\|_{q^*}.$$

In cases where $\rho\|\beta\|_{q^*}$ is small and its coefficient is close to one, robustification over features and regularization of logistic regression are approximately equivalent.

3.4.4 Optimal Decision Trees

Robustifying Problem (3.6) against the uncertainty set $\mathcal{U}_x$ gives a problem identical to Problem (3.6) except with the following constraints in place of the constraints (3.6p) and (3.6q):

$$\mathbf{a}_j^T(\mathbf{x}_i + \mathbf{\Delta x}_i) + \epsilon \leq b_j + M\,(1 - z_{ik}) \quad \forall \mathbf{\Delta X} \in \mathcal{U}_x, i = 1, \ldots, n, k = 1, \ldots, K, \forall j \in \mathcal{P}_k^l, \tag{3.16a}$$

$$\mathbf{a}_j^T(\mathbf{x}_i + \mathbf{\Delta x}_i) \geq b_j + M\,(1 - z_{ik}) \qquad \forall \mathbf{\Delta X} \in \mathcal{U}_x, i = 1, \ldots, n, k = 1, \ldots, K, \forall j \in \mathcal{P}_k^u. \tag{3.16b}$$

We refer to this optimization problem as Problem (3.16).

Theorem 3. *The robust counterpart to Problem* (3.16) *is identical to Problem* (3.16) *except with the following constraints in place of constraints* (3.16a) *and* (3.16b)*:*

$$\mathbf{a}_j^T\mathbf{x}_i + \rho + \epsilon \leq b_j + (1 - z_{ik}) \qquad i = 1, \ldots, n,\ k = 1, \ldots, K,\ \forall j \in \mathcal{P}_k^l, \tag{3.17a}$$

$$\mathbf{a}_j^T\mathbf{x}_i - \rho \geq b_j + (1 - z_{ik}) \qquad i = 1, \ldots, n,\ k = 1, \ldots, K,\ \forall j \in \mathcal{P}_k^l. \tag{3.17b}$$

Proof. Proof. Because constraint (3.16a) must hold for all $\mathbf{\Delta X} \in \mathcal{U}_x$, this constraint is equivalent to

$$\max_{\mathbf{\Delta X} \in \mathcal{U}_x} \left\{\mathbf{a}_j^T \mathbf{\Delta x}_i\right\} \leq b_j + M\,(1 - z_{ik}) - \mathbf{a}_j^T\mathbf{x}_i - \epsilon \quad i = 1, \ldots, n,\ k = 1, \ldots, K,\ \forall j \in \mathcal{P}_k^l.$$

This maximization term is equal to the optimal value of the following problem:

$$\begin{aligned} \max \quad & \mathbf{a}_j^T \mathbf{\Delta x}_i \\ \text{s.t.} \quad & \|\mathbf{\Delta x}_i\|_q \leq \rho. \end{aligned}$$

We recognize this as the dual norm problem, and by (3.7), the optimal value is

$\rho\|\mathbf{a}_j\|_{q^*}$, where ℓ_{q^*} is the dual norm of ℓ_q. Moreover, if this constraint is to be non-trivial (which requires $z_{ik} = 1$), we know from (3.6m) that $d_j = 0$ for all ancestors $j \in \mathcal{P}_k^l$. Thus, from (3.6j) we have that $\sum_l \mathbf{a}_{jl} = 1$ and so together with (3.6r) we know that a single element of $\mathbf{a}_j$ is 1 with all other elements being 0. This means that $\|\mathbf{a}_j\|_{q^*} = 1$ for any q, so the value of the maximization term is simply ρ. Rearranging terms yields the constraint (3.17a). We use an identical approach to yield (3.17b) from (3.16b). □

This remains a linear mixed-integer optimization problem regardless of the original choice of q. The only difference compared to the nominal problem is the introduction of a margin of size ρ around each b_j. The problem is therefore practically solvable like the nominal problem.

3.5 Robustness Against Uncertainty in Labels

In this section, we introduce the notion of robustifying classification methods against uncertainties in the labels of the training set. We consider a discrete uncertainty set which limits the number of incorrect labels to be less than or equal to a fixed number Γ. We then define and state the *label-robust counterpart* for each of the classification methods.

3.5.1 Motivating Label-Robustness

Uncertainties in data labels can occur naturally from errors in manual entries, self-reporting, or non-exact, non-objective label definition. To model uncertainty in the labels of the training set, we consider a scenario where some number of the supplied labels are incorrect. We introduce variables $\Delta y_i \in \{0, 1\}$, where 1 indicates that the label was incorrect and has in fact been flipped, and 0 indicates that the label was correct. We consider the following uncertainty set:

$$\mathcal{U}_y = \left\{ \mathbf{\Delta y} \in \{0,1\}^n \;\middle|\; \sum_{i=1}^{n} \Delta y_i \leq \Gamma \right\},$$

where Γ is an integer-valued parameter controlling the number of data points that we allow to be mislabeled. Observe that in contrast to the uncertainty set over the features, $\mathcal{U}_y$ cannot be decomposed as the Cartesian product of smaller uncertainty sets.

We can then model the true labels of the training set as $y_i(1-2\Delta y_i), i = 1, \ldots, n$. Applying robust optimization, we modify the training process so that our classifier is optimized against the worst-case realization $\mathbf{\Delta y} \in \mathcal{U}_y$ to obtain a classifier that is *label-robust*. In practice, the parameter Γ which determines the size of our uncertainty set is often modeled as a proportion of the total number of data points, and can be chosen via validation. Note that when $\Gamma = 0$ the problem is the same as the nominal problem. In this sense, our validation can include the nominal case, so the best label-robust solution will only be preferred over the nominal case if it leads to an improvement in accuracy in validation.

As in Section 3.4, we present the reformulated robust counterparts below for logistic regression, SVM, and optimal trees. For each method, we refer to the resulting deterministic optimization problem as the *label-robust counterpart* of that classifier.

3.5.2 Soft-Margin Support Vector Machines

Robustifying Problem (3.2) against the uncertainty set $\mathcal{U}_y$ gives

$$\min_{\mathbf{w},b} \max_{\Delta \mathbf{y} \in \mathcal{U}_y} \sum_{i=1}^{n} \max\{1 - y_i(1-2\Delta y_i)(\mathbf{w}^T\mathbf{x}_i - b), 0\}. \tag{3.18}$$

Theorem 4. *The robust counterpart to Problem* (3.18) *is*

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} \xi_i + \Gamma q + \sum_{i=1}^{n} r_i \\
\text{s.t.} \quad & q + r_i \geq \phi_i - \xi_i && i = 1, \ldots, n, \\
& \xi_i \geq 1 - y_i(\mathbf{w}^T \mathbf{x}_i - b) && i = 1, \ldots, n, \\
& \xi_i \leq 1 - y_i(\mathbf{w}^T \mathbf{x}_i - b) + M(1 - s_i) && i = 1, \ldots, n, \\
& \xi_i \leq M s_i && i = 1, \ldots, n, \\
& \phi_i \geq 1 + y_i(\mathbf{w}^T \mathbf{x}_i - b) && i = 1, \ldots, n, \\
& \phi_i \leq 1 + y_i(\mathbf{w}^T \mathbf{x}_i - b) + M(1 - t_i) && i = 1, \ldots, n, \\
& \phi_i \leq M t_i && i = 1, \ldots, n, \\
& r_i, \xi_i, \phi_i \geq 0 && i = 1, \ldots, n, \\
& q \geq 0, \\
& \mathbf{s}, \mathbf{t} \in \{0, 1\}^n.
\end{aligned} \tag{3.19}$$

where M is a sufficiently large constant.

Proof. Proof. Fix $\mathbf{w}$ and b, and consider the inner maximization problem

$$\max_{\mathbf{\Delta y} \in \mathcal{U}_y} \sum_{i=1}^{n} \max\{1 - y_i(1 - 2\Delta y_i)(\mathbf{w}^T \mathbf{x}_i - b), 0\} \quad i = 1, \ldots, n. \tag{3.20}$$

Define the functions

$$f_i(\Delta y_i) = \max\{1 - y_i(1 - 2\Delta y_i)(\mathbf{w}^T \mathbf{x}_i - b), 0\}, \quad i = 1, \ldots, n.$$

Since $\Delta y_i \in \{0, 1\}$ for all i, we observe

$$f_i(\Delta y_i) = [f_i(1) - f_i(0)]\Delta y_i + f_i(0) \quad i = 1, \ldots, n.$$

Let $\phi_i = f_i(1)$ and $\xi_i = f_i(0)$ for $i = 1, \ldots, n$. It follows that Problem (3.20) is equivalent to

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} (\phi_i - \xi_i)\Delta y_i + \xi_i \\ \text{s.t.} \quad & \mathbf{\Delta y} \in \mathcal{U}_y. \end{aligned}$$

Next, consider the following polyhedron, which is the convex hull of $\mathcal{U}_y$:

$$\mathcal{P}_y = \left\{ \mathbf{\Delta y} \in \mathbb{R}^n \;\middle|\; 0 \leq \Delta y_i \leq 1, \sum_{i=1}^{n} \Delta y_i \leq \Gamma \right\}.$$

Since the polyhedron $\mathcal{P}_y$ has integer extreme points, this problem is equivalent to its linear relaxation

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} (\phi_i - \xi_i)\Delta y_i + \xi_i \\ \text{s.t.} \quad & 0 \leq \Delta y_i \leq 1 && i = 1, \ldots, n, \\ & \sum_{i=1}^{n} \Delta y_i \leq \Gamma. \end{aligned}$$

By strong duality, this has the same objective value as its dual problem

$$\begin{aligned} \min \quad & \Gamma q + \sum_{i=1}^{n} r_i + \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & q + r_i \geq \phi_i - \xi_i && i = 1, \ldots, n, \\ & r_i \geq 0 && i = 1, \ldots, n, \\ & q \geq 0. \end{aligned}$$

Minimizing over $\mathbf{w}$ and b, this optimization problem becomes

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} \xi_i + \Gamma q + \sum_{i=1}^{n} r_i \\
\text{s.t.} \quad & q + r_i \geq \phi_i - \xi_i && i = 1, \ldots, n, \\
& \xi_i = \max\{1 - y_i(\mathbf{w}^T \mathbf{x}_i - b), 0\} && i = 1, \ldots, n, \\
& \phi_i = \max\{1 + y_i(\mathbf{w}^T \mathbf{x}_i - b), 0\} && i = 1, \ldots, n, \\
& r_i \geq 0 && i = 1, \ldots, n, \\
& q \geq 0.
\end{aligned}$$

Reformulating the problem to specify the values of the variables ξ_i, ϕ_i with linear constraints yields the desired result. □

Problem (3.19) is a mixed-integer optimization problem which is practically solvable.

3.5.3 Logistic Regression

Robustifying Problem (3.4) against the uncertainty set $\mathcal{U}_y$ gives

$$\max_{\beta, \beta_0} \min_{\Delta \mathbf{y} \in \mathcal{U}_y} -\sum_{i=1}^{n} \log\left(1 + e^{-y_i(1 - 2\Delta y_i)(\beta^T \mathbf{x}_i + \beta_0)}\right). \tag{3.21}$$

Theorem 5. *The robust counterpart to Problem* (3.21) *is*

$$\begin{aligned}
\max_{\beta, \beta_0} \quad & -\sum_{i=1}^{n} \log\left(1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}\right) + \Gamma \mu + \sum_{i=1}^{n} \nu_i \\
\text{s.t.} \quad & \mu + \nu_i \leq \log\left(\frac{1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}}{1 + e^{y_i(\beta^T \mathbf{x}_i + \beta_0)}}\right) && i = 1, \ldots, n, \\
& \nu_i \leq 0 && i = 1, \ldots, n, \\
& \mu \leq 0.
\end{aligned} \tag{3.22}$$

Proof. Proof. Define the functions $f_i(\Delta y_i) = -\log\left(1 + e^{-y_i(1-2\Delta y_i)(\beta^T \mathbf{x}_i + \beta_0)}\right)$ for $i = 1, \ldots, n$. Because $\Delta y_i \in \{0, 1\}$, we can express $f_i(\Delta y_i)$ as

$$\begin{aligned} f_i(\Delta y_i) &= [f(1) - f(0)]\Delta y_i + f(0) \\ &= \log\left(\frac{1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}}{1 + e^{y_i(\beta^T \mathbf{x}_i + \beta_0)}}\right)\Delta y_i - \log\left(1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}\right). \end{aligned}$$

We can thus rewrite the inner minimization part of Problem (3.21) as

$$\min_{\Delta \mathbf{y} \in \mathcal{U}_y} \quad \sum_{i=1}^{n} \left[\log\left(\frac{1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}}{1 + e^{y_i(\beta^T \mathbf{x}_i + \beta_0)}}\right)\Delta y_i - \log\left(1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}\right)\right]. \tag{3.23}$$

Since the convex hull of $\mathcal{U}_y$ has integer extreme points, Problem (3.23) has the same objective as its linear optimization relaxation [20]

$$\begin{aligned} \min_{\Delta \mathbf{y}} \quad & \sum_{i=1}^{n} \left[\log\left(\frac{1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}}{1 + e^{y_i(\beta^T \mathbf{x}_i + \beta_0)}}\right)\Delta y_i - \log\left(1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}\right)\right] \\ \text{s.t.} \quad & 0 \leq \Delta y_i \leq 1 \qquad\qquad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} \Delta y_i \leq \Gamma. \end{aligned} \tag{3.24}$$

By strong duality, the optimal value to Problem (3.24) is equal to that of its dual problem

$$\begin{aligned} \max \quad & -\sum_{i=1}^{n} \log\left(1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}\right) + \Gamma\mu + \sum_{i=1}^{n} \nu_i \\ \text{s.t.} \quad & \mu + \nu_i \leq \log\left(\frac{1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}}{1 + e^{y_i(\beta^T \mathbf{x}_i + \beta_0)}}\right) \qquad i = 1, \ldots, n, \\ & \nu_i \leq 0 \qquad\qquad i = 1, \ldots, n, \\ & \mu \leq 0. \end{aligned}$$

Substituting this back into Problem (3.21) in place of the inner minimization, it becomes a single maximization problem, giving the stated result. □

This problem has a twice continuously differentiable concave objective function and constraints, making it tractably solvable with an interior point method [4].

3.5.4 Optimal Decision Trees

Robustifying Problem (3.6) against the uncertainty set $\mathcal{U}_y$ gives a problem identical to Problem (3.6) with the following constraints in place of constraints (3.6b), (3.6c), (3.6d), (3.6e), (3.6f), and (3.6g):

$$g_k = \sum_{i=1}^{n} \frac{1 - y_i(1 - 2\Delta y_i)}{2} z_{ik} \qquad k = 1, \ldots, K, \tag{3.25a}$$

$$h_k = \sum_{i=1}^{n} \frac{1 + y_i(1 - 2\Delta y_i)}{2} z_{ik} \qquad k = 1, \ldots, K, \tag{3.25b}$$

$$f_k \leq g_k + M[w_k + (1 - c_k)] \qquad \forall \boldsymbol{\Delta y} \in \mathcal{U}_y,\ k = 1, \ldots, K, \tag{3.25c}$$

$$f_k \leq h_k + M[(1 - w_k) + (1 - c_k)] \qquad \forall \boldsymbol{\Delta y} \in \mathcal{U}_y,\ k = 1, \ldots, K, \tag{3.25d}$$

$$f_k \geq g_k - M[(1 - w_k) + (1 - c_k)] \qquad \forall \boldsymbol{\Delta y} \in \mathcal{U}_y,\ k = 1, \ldots, K, \tag{3.25e}$$

$$f_k \geq h_k - M[w_k + (1 - c_k)] \qquad \forall \boldsymbol{\Delta y} \in \mathcal{U}_y,\ k = 1, \ldots, K. \tag{3.25f}$$

We refer to this optimization problem as Problem (3.25).

Theorem 6. *The robust counterpart to Problem* (3.25) *is identical to Problem* (3.25) *with the following constraints in place of constraints* (3.25a), (3.25b), (3.25c),(3.25d), (3.25e), *and* (3.25f)*:*

$$g_k = \sum_{i=1}^{n} \frac{1 - y_i}{2} z_{ik} \qquad k = 1, \ldots, K, \tag{3.26a}$$

$$h_k = \sum_{i=1}^{n} \frac{1 + y_i}{2} z_{ik} \qquad k = 1, \ldots, K, \tag{3.26b}$$

$$f_k \leq g_k - \Gamma \mu_{1,k} - \sum_{i=1}^{n} \nu_{1,ik} + M[w_k + (1 - c_k)] \qquad k = 1, \ldots, K, \tag{3.26c}$$

$$f_k \leq h_k - \Gamma\mu_{2,k} - \sum_{i=1}^{n} \nu_{2,ik} + M[(1-w_k) + (1-c_k)] \qquad k = 1, \ldots, K, \tag{3.26d}$$

$$f_k \geq g_k + \Gamma\mu_{3,k} + \sum_{i=1}^{n} \nu_{3,ik} - M[(1-w_k) + (1-c_k)] \qquad k = 1, \ldots, K, \tag{3.26e}$$

$$f_k \geq h_k + \Gamma\mu_{4,k} + \sum_{i=1}^{n} \nu_{4,ik} - M[w_k + (1-c_k)] \qquad k = 1, \ldots, K, \tag{3.26f}$$

$$\mu_{m,k} + \nu_{m,ik} \geq -y_i z_{ik} \qquad i = 1, \ldots, n,\ k = 1, \ldots, K,\ m = 1, 4, \tag{3.26g}$$

$$\mu_{m,k} + \nu_{m,ik} \geq y_i z_{ik} \qquad i = 1, \ldots, n,\ k = 1, \ldots, K,\ m = 2, 3, \tag{3.26h}$$

$$\mu_{m,k}, \nu_{m,ik} \geq 0 \qquad i = 1, \ldots, n,\ k = 1, \ldots, K,\ m = 1, \ldots, 4. \tag{3.26i}$$

Proof. Proof. We can substitute (3.25a) into constraint (3.25c) to obtain

$$\sum_{i=1}^{n} \frac{1 - y_i(1 - 2\Delta y_i)}{2} z_{ik} \geq f_k - M[w_k + (1 - c_k)] \qquad \forall \mathbf{\Delta y} \in \mathcal{U}_y,\ k = 1, \ldots, K,$$

$$\sum_{i=1}^{n} \frac{1 - y_i}{2} z_{ik} + \sum_{i=1}^{n} y_i z_{ik} \Delta y_i \geq f_k - M[w_k + (1 - c_k)] \qquad \forall \mathbf{\Delta y} \in \mathcal{U}_y,\ k = 1, \ldots, K.$$

Since this must hold for all $\mathbf{\Delta y} \in \mathcal{U}_y$, this is equivalent to the following constraint:

$$\sum_{i=1}^{n} \frac{1 - y_i}{2} z_{ik} + \min_{\mathbf{\Delta y} \in \mathcal{U}_y} \left\{ \sum_{i=1}^{n} y_i z_{ik} \Delta y_i \right\} \geq f_k - M[w_k + (1 - c_k)] \quad k = 1, \ldots, K.$$

The convex hull of $\mathcal{U}_y$ has integer extreme points, so the value of the minimization term is equivalent to the optimal value of its linear relaxation (for any fixed k)

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} y_i z_{ik} \Delta y_i \\ \text{s.t.} \quad & 0 \leq \Delta y_i \leq 1 \qquad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} \Delta y_i \leq \Gamma. \end{aligned}$$

By strong duality, this problem has the same optimal objective value as its dual

$$\begin{aligned} \max \quad & \Gamma\mu_{1,k} + \sum_{i=1}^{n} \nu_{1,ik} \\ \text{s.t.} \quad & \mu_{1,k} + \nu_{1,ik} \leq y_i z_{ik} \qquad i = 1, \ldots, n, \\ & \mu_{1,k}, \nu_{1,ik} \leq 0 \qquad\qquad i = 1, \ldots, n. \end{aligned}$$

Substituting this back into the original constraint gives

$$\begin{aligned} & \sum_{i=1}^{n} \frac{1 - y_i}{2} z_{ik} + \Gamma\mu_{1,k} + \sum_{i=1}^{n} \nu_{1,ik} \geq f_k - M[w_k + (1 - c_k)] \qquad k = 1, \ldots, K, \\ & \mu_{1,k} + \nu_{1,ik} \leq y_i z_{ik} \qquad\qquad i = 1, \ldots, n, \\ & \mu_{1,k}, \nu_{1,ik} \leq 0 \qquad\qquad i = 1, \ldots, n. \end{aligned}$$

We substitute back for the original definition of g_k from (3.6b), and change the signs of μ and ν to get

$$\begin{aligned} & g_k - \Gamma\mu_{1,k} - \sum_{i=1}^{n} \nu_{1,ik} \geq f_k - M[w_k + (1 - c_k)] \qquad k = 1, \ldots, K, \\ & \mu_{1,k} + \nu_{1,ik} \geq -y_i z_{ik} \qquad\qquad i = 1, \ldots, n, \\ & \mu_{1,k}, \nu_{1,ik} \geq 0 \qquad\qquad i = 1, \ldots, n. \end{aligned}$$

We can rearrange this to obtain constraint (3.26c), as well as parts of constraints (3.26g) and (3.26i).

We repeat this entire process identically for constraints (3.25d), (3.25e), and (3.25f) to achieve the stated result. □

Similar to before, this remains a linear mixed-integer optimization problem, and so is practically solvable. The label-robustification for Optimal Decision Trees also has a simple geometric interpretation. Recall that in the model, g_k is the number of points in node k with label $y_i = +1$, h_k is the number of points in node k with label $y_i = -1$, and f_k is the number of points in node k that are misclassified, which in the nominal case is simply $\min\{g_k, h_k\}$. In the label-robust counterpart, the extra

terms in these constraints require feasible solutions to have strict separation between f_k, g_k and h_k. Indeed, we can obtain a feasible solution by setting $\mu_{m,k} = 1$ and $\nu_{m,ik} = 0$, which then requires $|g_k - h_k| \geq 2\Gamma$, and $f_k = \min\{g_k, h_k\} + \Gamma$. This means that a feasible label-robust solution requires the majority class in each node to be a strict majority, and the size of this required separation is controlled by the robustness parameter Γ. Increasing Γ has the effect of increasing the *label purity* of all nodes in the tree, since trees that do not have the required margin between g_k and h_k at every node k in the tree are treated as being infeasible for the label-robust problem.

3.6 Robustness in Both Features and Labels

In this section, we consider applying the methods of Sections 3.4 and 3.5 simultaneously to construct a new family of classifiers that are robust to uncertainty in both features and labels. We will refer to this family as *robust-in-both* classifiers. To develop these classifiers, we simply expose the classification problem to both feature-uncertainty with uncertainty set $\mathcal{U}_x$, and label-uncertainty with uncertainty set $\mathcal{U}_y$. This is a natural extension of our previous methods to handle classification problems which may have errors in both the features and labels of the training data. For example, in the contraceptive method choice data set considered in Section 3.5, survey data is used to obtain information on both the features (demographic and socio-economic characteristics) and labels (contraceptive method choice), and both factors may be influenced by inaccurate reporting.

We present the reformulated robust counterparts below for soft-margin support vector machines, logistic regression, and optimal decisions trees, which we refer to as the *robust-in-both counterpart* for each method. The proofs are similar to the derivations of the robust counterparts in the previous two sections, and are included in the end.

Like both methods individually, the robust-in-both classifier has to select the robustness parameters ρ and Γ through validation. As per the individual cases, when we set $\rho = \Gamma = 0$, the problem reduces to the nominal problem. Note also that if only

one of ρ/Γ is zero, the problem reduces to the label-robust/feature-robust problem respectively. This means that as part of the robust-in-both validation process, we consider the models from the nominal, feature-robust and label-robust classifiers in addition to the robust-in-both classifier, and then select the classifier among these with the best validation accuracy. In this sense, the robust-in-both classifier is the strongest of all the robust classifiers, since it selects in validation the best performing robust classifier of all those we have considered.

3.6.1 Soft-Margin Support Vector Machines

Robustifying Problem (3.1) against both $\mathcal{U}_x$ and $\mathcal{U}_y$ gives the following robust optimization problem:

$$\min_{\mathbf{w},b} \max_{\Delta\mathbf{y}\in\mathcal{U}_y} \max_{\Delta\mathbf{X}\in\mathcal{U}_x} \sum_{i=1}^{n} \max\{1 - y_i(1 - 2\Delta y_i)(\mathbf{w}^T(\mathbf{x}_i + \boldsymbol{\Delta}\mathbf{x}_i) - b), 0\}. \tag{3.27}$$

Theorem 7. *The robust counterpart to Problem* (3.27) *is*

$$\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{n} \xi_i + \Gamma q + \sum_{i=1}^{n} r_i & \\
\text{s.t.} & q + r_i \geq \phi_i - \xi_i & i = 1, \ldots, n, \\
& \xi_i \geq 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b) + \rho\|\mathbf{w}\|_{q^*} & i = 1, \ldots, n, \\
& \xi_i \leq 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b) + \rho\|\mathbf{w}\|_{q^*} + M(1 - s_i) & i = 1, \ldots, n, \\
& \xi_i \leq M s_i & i = 1, \ldots, n, \\
& \phi_i \geq 1 + y_i(\mathbf{w}^T\mathbf{x}_i - b) + \rho\|\mathbf{w}\|_{q^*} & i = 1, \ldots, n, \\
& \phi_i \leq 1 + y_i(\mathbf{w}^T\mathbf{x}_i - b) + \rho\|\mathbf{w}\|_{q^*} + M(1 - t_i) & i = 1, \ldots, n, \\
& \phi_i \leq M t_i & i = 1, \ldots, n, \\
& r_i, \xi_i, \phi_i \geq 0 & i = 1, \ldots, n, \\
& q \geq 0, & \\
& \mathbf{s}, \mathbf{t} \in \{0, 1\}^n. &
\end{array} \tag{3.28}$$

where ℓ_{q^*} is the dual norm of ℓ_q, and M is a sufficiently large constant.

The proof of Theorem 7 is straightforward as it is a direct result of applying the proofs for robustness in features and labels sequentially.

Problem (3.28) is a mixed-integer optimization problem which is practically solvable.

3.6.2 Logistic Regression

Robustifying Problem (3.4) against both $\mathcal{U}_x$ and $\mathcal{U}_y$ gives the following robust optimization problem:

$$\max_{\beta,\beta_0} \min_{\Delta\mathbf{y}\in\mathcal{U}_y} \min_{\Delta\mathbf{X}\in\mathcal{U}_x} -\sum_{i=1}^{n} \log\left(1+e^{-y_i(1-2\Delta y_i)(\beta^T(\mathbf{x}_i+\boldsymbol{\Delta}\mathbf{x}_i)+\beta_0)}\right). \tag{3.29}$$

Theorem 8. *The robust counterpart to Problem* (3.29) *is*

$$\begin{aligned}
\max \quad & -\sum_{i=1}^{n} \log\left(1+e^{-y_i(\beta^T\mathbf{x}_i+\beta_0)+\rho\|\beta\|_{q^*}}\right) + \Gamma\mu + \sum_{i=1}^{n}\nu_i \\
\text{s.t.} \quad & \mu+\nu_i \le \log\left(\frac{1+e^{-y_i(\beta^T\mathbf{x}_i+\beta_0)+\rho\|\beta\|_{q^*}}}{1+e^{y_i(\beta^T\mathbf{x}_i+\beta_0)+\rho\|\beta\|_{q^*}}}\right) && i=1,\ldots,n, \\
& \nu_i \le 0 && i=1,\ldots,n, \\
& \mu \le 0.
\end{aligned} \tag{3.30}$$

where ℓ_{q^*} is the dual norm of ℓ_q.

The proof of Theorem 8 essentially applies the process in the proof for feature-robust logistic regression, followed by the process in the proof for label-robustness to obtain the final robust counterpart.

Problem (3.30) is a maximization of a concave, twice continuously differentiable function in β and β_0 with constraints for any given ρ and Γ. Therefore, we can solve this problem using interior point methods [4].

3.6.3 Optimal Decision Trees

Robustifying Problem (3.6) against both $\mathcal{U}_x$ and $\mathcal{U}_y$ gives a problem identical to Problem (3.6) with the following exceptions:

- The constraints in (3.16) in place of constraints (3.6p) and (3.6q);
- The constraints in (3.25) in place of constraints (3.6b), (3.6c), (3.6d), (3.6e), (3.6f), and (3.6g).

Theorem 9. *The robust counterpart to the above problem is identical to Problem* (3.6) *with the following exceptions:*

- *The constraints in* (3.17) *in place of constraints* (3.6p) *and* (3.6q)*;*
- *The constraints in* (3.26) *in place of constraints* (3.6b)*,* (3.6c)*,* (3.6d)*,* (3.6e)*,* (3.6f)*, and* (3.6g)*.*

The proof of Theorem 9 is again a direct corollary to the proofs in robustness in features and robustnesss in labels. This resulting problem is still a linear mixed-integer optimization problem, and so remains practically solvable.

3.7 Computational Experiments with Synthetic Data Sets

In this section, we evaluate the performance of robust methods on synthetically-generated data sets in order to understand the relative performance of the different types of robustness and also how robust methods compare to the regularized methods used in practice. In these experiments, we run SVM and logistic regression methods to recover the separating hyperplane classifier on a synthetic example. We focus on SVM and logistic regression in this analysis because both of these classification models are suitable given the data generation process and have widely used regularized methods to compare against.

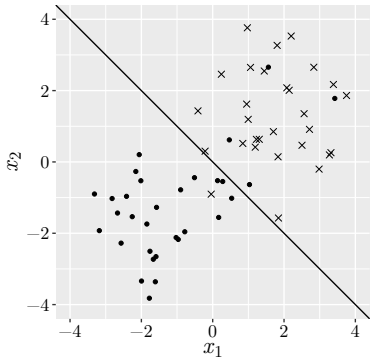


Figure 3-2: Example of synthetically-generated data in two dimensions alongside the true generating hyperplane.

3.7.1 Experimental Setup

The experiment uses data in $\mathbb{R}^2$. The data is generated synthetically in three parts:

1. 25 points are generated as multivariate random normal, $N(1.5\mathbf{e}, \mathbf{I})$, where $\mathbf{e}$ is the vector of ones and $\mathbf{I}$ is the identity matrix. These points are given the label $+1$.

2. 25 points are generated as multivariate random normal, $N(-1.5\mathbf{e}, \mathbf{I})$ and labeled -1.

3. 10 outlier points are introduced as multivariate random normal, $N(\mathbf{0}, 3\mathbf{I})$, where $\mathbf{0}$ is the vector of zeros. The labels are randomly generated as either -1 or $+1$.

We split this data 75%/25% into training and validation sets, which we used to tune the parameters for the regularized and robust methods. We included relatively few points in the training and validation sets to make the classification task nontrivial given the simple data generation process. To create the test set, we generated 10,000 points in the same way as each major cluster of points (items 1 and 2 above).

An example of a data set generated according to this procedure is shown in Figure 3-2. We can see that there are two distinct clusters of points, with some scattered noise centered in the area between the two clusters. By the symmetry of this data generation process, we can see that the true hyperplane separating the two clusters of points is given by the equation $\mathbf{e}^T\mathbf{x} = 0$, also shown in Figure 3-2. The goal of the

experiment is to determine how closely the various methods can recover this truth in the data in the presence of added noise via the addition of these outlier points. In particular, we are interested in the following two measures:

- *Accuracy:* We measure accuracy by reporting the out-of-sample error of the trained classifiers on the larger test set.
- *Similarity:* To evaluate the ability of each method to recover the truth in the data, we measure the norm of the difference between the separating hyperplane generated by the methods and the true hyperplane ($\mathbf{e}^T\mathbf{x} = 0$).

3.7.2 Classification Methods

For these experiments, we consider SVM and logistic regression, as these both create classifiers with a single hyperplane, which matches the truth in the synthetic data. In both cases, we compare the nominal method, the regularized method, and all three robust methods (features, labels and both). Each method was implemented in the Julia programming language, a rapidly maturing language designed for high-performance scientific computing [22]. The optimization problems required by each method were formulated in JuMP, a state-of-the-art library for algebraic modeling and mathematical optimization [71]. The commercial solver Gurobi [53] was used to solve the linear and mixed-integer optimization problems for SVM, and the open source solver Ipopt [94] was used to solve the convex optimization problems for logistic regression.

To ensure a fair comparison, we use the ℓ_1 norm in the regularized methods and set $q = \infty$ for the feature uncertainty set so that the norms in the robust methods are also ℓ_1 norms. For each method, the values of ρ and Γ were selected through validation when using the corresponding robust classifiers.

3.7.3 Results

This experiment was repeated 2000 times, and we present the means and standard errors of the two measures for each method in Table 3.1. For SVM, the nominal and

regularized methods have roughly the same power in recovering the truth in the data, after accounting for the standard errors. The feature-robust method improves upon the nominal method in both measures, and the label-robust method further improves upon both measures. The best performance in both measures is obtained when we consider both types of robustness simultaneously in the robust-in-both method, and this method improves significantly upon both methods that consider only a single type of robustness.

For logistic regression, we see that the nominal method performs the worst in both measures. The regularized method and our feature-robust method are roughly comparable, with the regularized method having a slight edge, and both offering a small improvement over the nominal method. As with SVM, the label-robust method offers significant improvement in both measures, and the robust-in-both method adds a further slight improvement on top of label-robust, showing that considering both types of robustness leads to additional power over considering just a single type.

In Table 3.2, we break down the results by percentile in out-of-sample error, and we report the $10^{th}, 20^{th}, \ldots, 90^{th}$ percentiles for each method. We find that robust methods match or outperform nominal and regularized methods across the board, and this relative improvement increases as the percentile increases. This follows our expectation that these robust methods reliably produce high quality classifiers, which protects us from giving biased predictions in worst case scenarios. In the worst

Table 3.1: Performance results for synthetic data experiments. For each method, we report the mean and standard error over 2000 runs for both the out-of-sample error and the distance of the generated classifier from the truth in the data.

Method	SVM		Logistic Regression	
	Out-of-sample error (%)	Distance from truth	Out-of-sample error (%)	Distance from truth
Nominal	2.571 ± 0.021	0.357 ± 0.004	2.717 ± 0.023	0.388 ± 0.004
Regularized	2.643 ± 0.027	0.357 ± 0.004	2.694 ± 0.022	0.384 ± 0.004
Features	2.516 ± 0.020	0.345 ± 0.004	2.701 ± 0.023	0.386 ± 0.004
Labels	2.396 ± 0.018	0.320 ± 0.004	2.450 ± 0.019	0.332 ± 0.004
Both	2.363 ± 0.018	0.310 ± 0.004	2.436 ± 0.019	0.329 ± 0.004

Table 3.2: Out-of-sample error results by percentile for synthetic data experiments.

Classifier	Method	Percentile				
		90^{th}	70^{th}	50^{th}	30^{th}	10^{th}
SVM	Nominal	3.771	2.695	2.275	1.985	1.774
	Regularized	3.941	2.700	2.235	1.975	1.775
	Features	3.651	2.650	2.225	1.965	1.755
	Labels	3.381	2.460	2.125	1.915	1.755
	Both	3.325	2.425	2.100	1.890	1.740
Logistic regression	Nominal	4.096	2.940	2.400	2.050	1.795
	Regularized	4.041	2.910	2.385	2.045	1.795
	Features	4.050	2.917	2.393	2.043	1.790
	Labels	3.552	2.565	2.175	1.928	1.745
	Both	3.515	2.550	2.165	1.920	1.745

case scenario presented (90^{th} percentile out-of-sample error), robust-in-both SVM and logistic regression yield out-of-sample errors of 3.325% and 3.515%, while regularized methods give out-of-sample errors of 3.941% and 4.041%, respectively.

From these experiments on synthetic data, we conclude that our robust classifiers can effectively deal with data that has been contaminated with noise. For both SVM and logistic regression, we observe that the robust methods offer significant improvements over the nominal and regularized methods, both in their accuracy and in their ability to correctly recover the truth in the data. Further, we found that the robust-in-both methods which combine robustness in the features and labels performed stronger than the feature-robust and label-robust methods individually, demonstrating that there is value in considering both types of uncertainty simultaneously.

3.8 Computational Experiments with Real-world Data Sets

In this section, we report on a series of comprehensive computational benchmarks to compare robust methods to their nominal counterparts. We also explore problem characteristics which influence the performance gain of robust methods, and derive a

simple decision rule recommending when robust classification should be applied.

3.8.1 Experimental Setup

In order to comprehensively report performance of the robust classification methods on real data sets, we tested the accuracy of these methods on a selection of 75 problems from the UCI Machine Learning Repository [66]. The data sets were selected to give a variety of problem sizes and difficulties to form a representative sample of real-world problems, with the largest data set having $n = 245,057$ observations, and the highest number of features being $p = 857$.

To obtain a binary classification problem for each data set, we consider the *one-vs.-rest* problem of predicting the occurrence of the first class in the data set. Each data set was split into three parts: the training set (60%), the validation set (20%) and the testing set (20%). The training set was used to train each classifier for a variety of combinations of input parameters. For each combination of parameters, the misclassification error on the validation set was calculated, and this was used to select the best combination of parameters for each classifier. Finally, the classifier was trained using these best parameters on the combined training and validation sets, before reporting the out-of-sample misclassification error on the testing set. All methods were trained, validated, and tested on the same random splits, and computational experiments were repeated five times for each data set with different splits. For each data set and classification method we report the average out-of-sample accuracy across all five splits.

3.8.2 Classification Methods

In these real-world experiments, we consider all three classification methods: SVM, logistic regression, and decision trees. We set $q = \infty$ for all of the feature-robust and robust-in-both uncertainty sets, so that all the norms in the robust methods are ℓ_1. The implementations for SVM and logistic regression are identical to those used in the synthetic experiments, which are described in Section 3.7.1. We implement Optimal

Table 3.3 Out-of-sample accuracy averaged across five seeds for each classification method and its robust counterparts on all data sets.

Data Set Information			SVM				Logistic Regression				CART			
UCI Data Set Name	n	p	Nominal	Features	Labels	Both	Nominal	Features	Labels	Both	Nominal	Features	Labels	Both
acute-inflammations-1	120	7	**1.0000**	**1.0000**	**1.0000**	0.9083	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9583	**1.0000**	**1.0000**	**1.0000**
acute-inflammations-2	120	7	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9750	**1.0000**	0.9833	**1.0000**
arrhythmia	68	280	0.5692	**0.7077**	0.6923	0.6308	**0.6923**	**0.6923**	**0.6923**	**0.6923**	0.6769	0.7077	0.6923	**0.7692**
balance-scale	625	5	**0.9200**	**0.9200**	**0.9200**	**0.9200**	**0.9200**	**0.9200**	**0.9200**	**0.9200**	**0.9200**	**0.9200**	**0.9200**	**0.9200**
banknote-authentication	1372	5	0.9912	0.9869	**0.9927**	0.9912	0.9920	0.9905	**0.9927**	0.9905	0.9533	**0.9642**	0.9635	0.9635
blood-transfusion	748	5	**0.7638**	**0.7638**	**0.7638**	**0.7638**	**0.7826**	0.7812	0.7785	0.7812	0.7799	**0.7893**	0.7799	0.7799
breast-cancer	683	10	0.9500	**0.9574**	0.9559	0.9559	0.9559	**0.9574**	0.9544	**0.9574**	0.9338	**0.9456**	0.9426	0.9426
breast-cancer-diagnostic	569	31	0.9351	0.9596	0.9439	**0.9614**	0.9561	**0.9684**	0.9526	**0.9684**	0.9281	0.9053	**0.9333**	0.9018
breast-cancer-prognostic	194	33	0.7128	0.7128	0.7128	**0.7179**	0.7128	0.7231	**0.7692**	0.7590	0.7436	0.7385	0.7436	**0.7538**
car-evaluation	1728	16	**0.8000**	0.7930	0.7832	0.7826	**0.8006**	0.7925	0.7994	0.7925	**0.8603**	0.8551	0.8597	0.8597
chess-king-rook-vs-king	28056	35	**0.9004**	**0.9004**	**0.9004**	**0.9004**	**0.9004**	**0.9004**	**0.9004**	**0.9004**	**0.9004**	**0.9004**	**0.9004**	**0.9004**
chess-king-rook-vs-king-pawn	3196	38	**0.9743**	0.9731	**0.9743**	0.9687	**0.9756**	0.9750	0.9734	0.9743	**0.9693**	**0.9693**	**0.9693**	**0.9693**
climate-model-crashes	540	19	0.9500	**0.9593**	0.9537	0.9574	0.9500	0.9537	**0.9556**	0.9537	0.9259	0.9241	**0.9296**	0.9241
cnae-9	1080	857	0.9750	**0.9861**	0.9685	0.9481	0.9806	0.9796	0.9796	**0.9824**	0.9657	**0.9722**	0.9704	0.9694
congressional-voting-records	232	17	0.9565	**0.9870**	0.9783	0.9826	0.9739	**0.9826**	0.9739	**0.9826**	**0.9870**	0.9826	**0.9870**	**0.9870**
connectionist-bench	990	11	0.9758	0.9758	**0.9778**	0.9768	0.9747	**0.9778**	0.9737	0.9768	**0.9747**	0.9737	0.9727	0.9727
connectionist-bench-sonar	208	61	0.7073	**0.7707**	0.7561	0.7561	0.7463	0.7512	**0.7756**	0.7659	0.7268	**0.7317**	0.7122	0.7317
contraceptive-method-choice	1473	12	0.6776	0.6769	**0.6789**	0.6755	0.6714	0.6769	0.6748	**0.6776**	0.6891	0.6980	**0.6986**	**0.6986**
credit-approval	653	38	0.8508	0.8569	0.8492	**0.8585**	0.8615	0.8615	0.8600	**0.8631**	**0.8569**	0.8415	0.8554	0.8415
cylinder-bands	277	485	0.5564	**0.7164**	0.5891	0.6691	**0.6727**	**0.6727**	**0.6727**	**0.6727**	0.6764	0.6800	0.6691	**0.7018**
dermatology	358	35	0.9662	0.9887	**0.9972**	0.9803	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9887**	**0.9887**	0.9859	**0.9887**
echocardiogram	61	7	**0.7167**	0.7000	0.6833	0.6833	**0.7833**	0.7500	**0.7833**	0.7333	**0.7500**	0.7167	0.7333	**0.7500**
ecoli	336	8	0.9582	0.9522	0.9582	**0.9582**	**0.9612**	**0.9612**	**0.9612**	0.9582	**0.9493**	0.9343	0.9284	0.9284
fertility	100	13	0.8700	**0.9000**	0.8800	**0.9000**	0.8700	**0.9000**	0.8800	**0.9000**	**0.9000**	0.8400	0.8900	0.8400
flags	194	60	0.6923	**0.8769**	0.7949	0.8205	0.7641	**0.8564**	0.8462	**0.8564**	0.8821	0.8872	0.8923	**0.9026**

For each data set, the best result (or multiple in the case of ties) for each method is indicated in bold, and the best method overall for the data set is underlined.

Table 3.3 (Cont.) Out-of-sample accuracy averaged across five seeds for each classification method and its robust counterparts on all data sets.

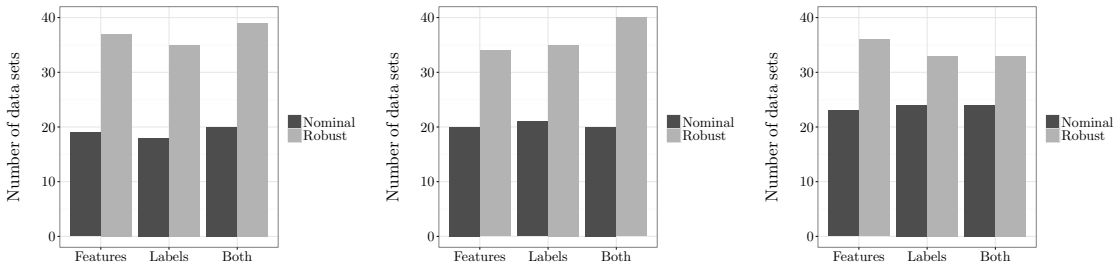
Data Set Information			SVM				Logistic Regression				CART			
UCI Data Set Name	n	p	Nominal	Features	Labels	Both	Nominal	Features	Labels	Both	Nominal	Features	Labels	Both
glass-identification	214	10	0.7163	0.7070	**0.7395**	0.7256	0.7116	0.7209	**0.7488**	0.7349	0.7674	0.7814	**0.7860**	**0.7860**
haberman-survival	306	4	0.7279	**0.7344**	**0.7344**	**0.7344**	**0.7410**	0.7311	0.7344	0.7311	**0.7049**	0.6623	0.6820	0.6787
hayes-roth	132	5	**0.6846**	0.6846	0.6769	0.6692	0.6615	**0.8000**	0.6769	0.7923	0.8154	**0.8154**	0.7385	0.7385
heart-disease-cleveland	297	19	**0.8407**	0.8339	0.8339	0.8203	0.8305	0.8271	**0.8339**	0.8305	0.7559	0.8000	0.7695	**0.8068**
hepatitis	80	20	**0.8500**	**0.8500**	0.8000	0.8125	0.8375	0.8250	**0.8625**	0.8250	0.8125	0.7875	**0.8250**	0.7875
hill-valley	606	101	0.5884	**0.9620**	0.5884	**0.9620**	**0.9934**	0.9636	0.9421	0.9636	**0.5504**	**0.5504**	**0.5504**	**0.5504**
hill-valley-noise	606	101	0.8612	0.8545	**0.8628**	0.8512	0.8463	**0.8876**	0.8083	**0.8876**	0.4744	**0.4992**	0.4942	0.4959
image-segmentation	210	20	0.9286	**0.9857**	0.9667	0.9476	0.9762	**0.9762**	**0.9762**	**0.9762**	0.9476	**0.9810**	0.9714	**0.9810**
indian-liver-patient	579	11	**0.7155**	**0.7155**	**0.7155**	**0.7155**	0.7172	0.7155	**0.7224**	**0.7224**	**0.6931**	0.6862	0.6914	0.6845
ionosphere	351	35	**0.8743**	**0.8743**	0.8514	**0.8743**	**0.8829**	0.8743	0.8571	0.8714	0.8971	**0.9086**	0.8914	**0.9086**
iris	150	5	**1.0000**	**1.0000**	**1.0000**	0.9800	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9867	**1.0000**
letter-recognition	20000	17	0.9921	0.9922	**0.9923**	**0.9923**	0.9907	0.9907	0.9907	**0.9908**	**0.9896**	**0.9896**	**0.9896**	**0.9896**
libras-movement	360	91	0.9056	**0.9694**	0.9611	**0.9694**	0.9444	**0.9639**	0.9528	**0.9639**	0.9333	0.9361	**0.9528**	0.9389
magic-gamma-telescope	19020	11	0.7922	0.7923	**0.7924**	**0.7924**	0.7916	0.7919	**0.7920**	0.7919	0.8364	0.8364	**0.8367**	**0.8367**
mammographic-mass	830	11	**0.8193**	0.8072	0.8000	0.8060	**0.8289**	**0.8289**	0.8217	0.8217	**0.8289**	0.8145	0.8217	0.8108
monks-problems-1	124	12	**0.8240**	0.7360	0.8000	0.8000	0.7440	0.7680	0.7760	**0.7920**	0.8080	**0.8400**	**0.8400**	**0.8400**
monks-problems-2	169	12	0.6118	**0.6176**	0.6118	**0.6176**	0.5647	**0.6235**	0.6176	**0.6235**	0.6118	0.6294	**0.6353**	**0.6353**
monks-problems-3	122	12	0.9167	**0.9333**	0.9167	**0.9333**	0.8500	0.9250	**0.9333**	0.9250	0.8917	**0.9333**	**0.9333**	**0.9333**
mushroom	5644	77	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9894	**0.9901**	0.9894	0.9894
nursery	12960	20	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.7910	0.7910
optical-recognition	3823	65	0.9929	0.9961	**0.9974**	0.9966	0.9942	0.9969	**0.9976**	0.9971	**0.9830**	**0.9830**	**0.9830**	**0.9830**
ozone-level-detection-eight	1847	73	**0.9301**	0.9301	0.9301	0.9295	0.9312	**0.9322**	0.9295	0.9317	0.9322	0.9322	**0.9328**	**0.9328**
ozone-level-detection-one	1848	73	0.9545	**0.9702**	0.9561	**0.9702**	0.9561	**0.9702**	0.9686	**0.9702**	**0.9702**	**0.9702**	**0.9702**	**0.9702**
parkinsons	195	22	0.8564	0.8359	0.8513	**0.8615**	0.8410	0.8103	**0.8462**	0.8205	0.8615	0.8410	**0.8821**	0.8410
pen-based-recognition	7494	17	**0.9904**	0.9889	0.9901	0.9891	**0.9903**	0.9899	0.9897	0.9897	0.9849	**0.9893**	0.9848	0.9848
pima-indians-diabetes	768	9	0.7765	0.7778	0.7765	**0.7791**	0.7778	0.7739	**0.7791**	0.7752	**0.7542**	0.7373	0.7477	0.7294

For each data set, the best result (or multiple in the case of ties) for each method is indicated in bold, and the best method overall for the data set is underlined.

Table 3.3 (Cont.) Out-of-sample accuracy averaged across five seeds for each classification method and its robust counterparts on all data sets.

Data Set Information			SVM				Logistic Regression				CART			
UCI Data Set Name	n	p	Nominal	Features	Labels	Both	Nominal	Features	Labels	Both	Nominal	Features	Labels	Both
planning-relax	182	13	**0.7222**	**0.7222**	**0.7222**	**0.7222**	0.6778	0.6944	0.6944	**0.7000**	**0.6889**	0.6833	**0.6889**	0.6556
poker-hand	25010	11	0.5010	0.5023	**0.5028**	0.5023	**0.5028**	0.5006	0.5018	0.5000	**0.5913**	**0.5913**	**0.5913**	**0.5913**
post-operative-patient	87	14	0.6235	0.6471	**0.7059**	**0.7059**	0.6118	0.6353	**0.6588**	**0.6588**	**0.6824**	0.6235	0.6471	0.6235
qsar-biodegradation	1055	42	0.8749	0.8758	**0.8777**	0.8758	**0.8730**	**0.8730**	0.8701	0.8682	0.7943	**0.8142**	0.8104	0.8114
seeds	210	8	0.9524	0.9429	**0.9524**	0.9429	**0.9571**	0.9524	0.9524	0.9476	0.8429	0.8762	0.8667	**0.9000**
seismic-bumps	2584	21	**0.9342**	**0.9342**	**0.9342**	**0.9342**	0.9319	**0.9327**	0.9327	0.9327	**0.9342**	**0.9342**	**0.9342**	**0.9342**
skin-segmentation	245057	4	0.9281	0.9348	0.9328	**0.9366**	0.9184	0.9184	0.9345	**0.9345**	**0.9656**	**0.9656**	**0.9656**	**0.9656**
soybean-large	266	63	0.7736	**0.8830**	0.8642	0.8717	0.7962	0.8792	**0.9019**	0.8868	**0.8830**	0.8642	0.8491	0.8491
soybean-small	47	38	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
spambase	4601	58	0.9257	**0.9265**	0.9252	**0.9265**	0.9230	**0.9248**	0.9246	0.9246	0.8896	0.8913	**0.8926**	**0.8926**
spect-heart	80	23	0.5875	**0.7125**	0.6500	**0.7125**	0.6750	**0.7875**	0.6625	0.7625	0.7625	**0.7750**	**0.7750**	**0.7750**
spectf-heart	80	45	0.5625	**0.6750**	0.6625	0.6500	0.5875	0.7500	0.5375	**0.7625**	0.7625	0.7250	**0.8000**	0.7375
statlog-project-german-credit	1000	49	0.7300	**0.7420**	0.7380	0.7400	0.7380	**0.7470**	0.7400	0.7400	**0.7250**	0.7100	0.7030	0.7150
statlog-project-landsat-satellite	4435	37	0.9811	0.9813	**0.9822**	0.9820	**0.9833**	0.9826	0.9833	0.9824	0.9477	0.9484	**0.9511**	**0.9511**
teaching-assistant-evaluation	151	53	**0.7000**	0.6733	0.6867	0.6733	**0.7133**	0.7067	**0.7133**	0.7067	0.6467	0.6733	0.6267	**0.7067**
thoracic-surgery	470	25	0.8426	**0.8511**	0.8426	**0.8511**	0.8213	**0.8489**	0.8362	0.8468	**0.8511**	0.8426	**0.8511**	0.8426
thyroid-disease-ann-thyroid	3772	22	0.9920	**0.9926**	0.9918	0.9915	0.9934	0.9934	**0.9936**	0.9934	0.9915	**0.9971**	**0.9971**	**0.9971**
thyroid-disease-new-thyroid	215	6	0.8977	0.8837	**0.8977**	0.8884	**0.8977**	**0.8977**	**0.8977**	**0.8977**	0.8884	0.9023	**0.9302**	0.9116
tic-tac-toe-endgame	958	19	**0.9801**	**0.9801**	**0.9801**	**0.9801**	**0.9801**	**0.9801**	**0.9801**	**0.9801**	0.9005	**0.9026**	0.8995	0.8995
wall-following-robot-2	5456	3	0.6220	**0.6544**	0.5415	0.5553	0.6081	0.6114	**0.6609**	**0.6609**	0.9879	**1.0000**	**1.0000**	**1.0000**
wall-following-robot-24	5456	5	0.6235	0.6544	0.6420	**0.6561**	0.6301	0.6348	**0.6565**	0.6563	0.9879	**1.0000**	**1.0000**	**1.0000**
wine	178	14	0.9657	0.9657	**0.9714**	0.9657	0.9714	0.9714	**0.9943**	**0.9943**	0.9257	**0.9429**	0.9371	0.9371
yeast	1484	9	**0.6902**	**0.6902**	**0.6902**	**0.6902**	0.6801	0.6828	**0.6929**	**0.6929**	**0.7286**	0.7219	0.7219	0.7219
zoo	101	17	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**

For each data set, the best result (or multiple in the case of ties) for each method is indicated in bold, and the best method overall for the data set is underlined.



(a) SVM (b) Logistic Regression (c) CART

Figure 3-3: Pairwise comparisons between nominal and robustness methods, showing the number of data sets for which each type of robustness generates the highest out-of-sample accuracy for each method.

Decision Trees using the JuMP software package in Julia, and the commercial solver Gurobi [53] was used to solve the mixed-integer optimization problems.

As in the other two methods, for Optimal Decision Trees we select the values of ρ and Γ through validation when using the corresponding robust classifier. During validation, we also select the complexity parameter (`cp`), the minimum number of points per node (`minbucket`), and the exploration depth around the warm start solution (`explorationdepth`). See [7] for a full description of these parameters. We compare the robust counterparts of the Optimal Decision Tree problem to the CART heuristic rather than the nominal Optimal Decision Tree problem. This allows us to provide a benchmark of the robust methods against the state-of-the-art methods that are widely used today. For the CART method we used the rpart package [89] in the R programming language [88].

Table 3.3 shows the out-of-sample accuracy performance of each classification method and its robust counterparts on all selected data sets. For each data set, the best result (or multiple in the case of ties) for each method is indicated in bold, and the best method overall for the data set is underlined.

3.8.3 Pairwise Comparisons

First, we present the results comparing individual robust classification methods against their nominal counterparts.

Table 3.4: Pairwise comparisons of robust classification methods against their nominal counterparts.

Nominal Method	Robustness Type	Wins	Losses	Ties
SVM	Features	37	19	19
	Labels	35	18	22
	Both	39	20	16
Logistic Regression	Features	34	20	21
	Labels	35	21	19
	Both	40	20	15
CART	Features	36	23	16
	Labels	33	24	18
	Both	33	24	18

Results for the three nominal methods and all robust variations are summarized in Figure 3-3. Each pair of bars in the graph represent a pairwise comparison between a specific robust method and its nominal counterpart. Each bar represents the number of data sets for which the either the robust or nominal method produced the single strongest classifier, based on out-of-sample accuracy. We see that for each classification method, all types of robustness have a lead over the nominal ones. In the case of logistic regression and SVM, robust-in-both produces most improvement in the number of correctly classified data sets. However for CART, it is the feature robust method that is most effective in improving classification over the nominal counterpart. Because the robust-in-both method encompasses the individual feature and label robust methods, this result could be due to difficulties in validation where the selected combination of robustness parameters did not lead to better out-of-sample performance than the individual robust methods. The exact counts of wins, ties, and losses for each robust counterpart compared to the corresponding nominal method are shown in Table 3.4.

Next, we consider the best of the nominal and robust-in-both methods across SVM, logistic regression, and CART. For each data set, we recorded which of these six methods had the highest out-of-sample accuracy. Figure 3-4 shows the breakdown of counts for data sets in which there is a unique highest out-of-sample accuracy. All

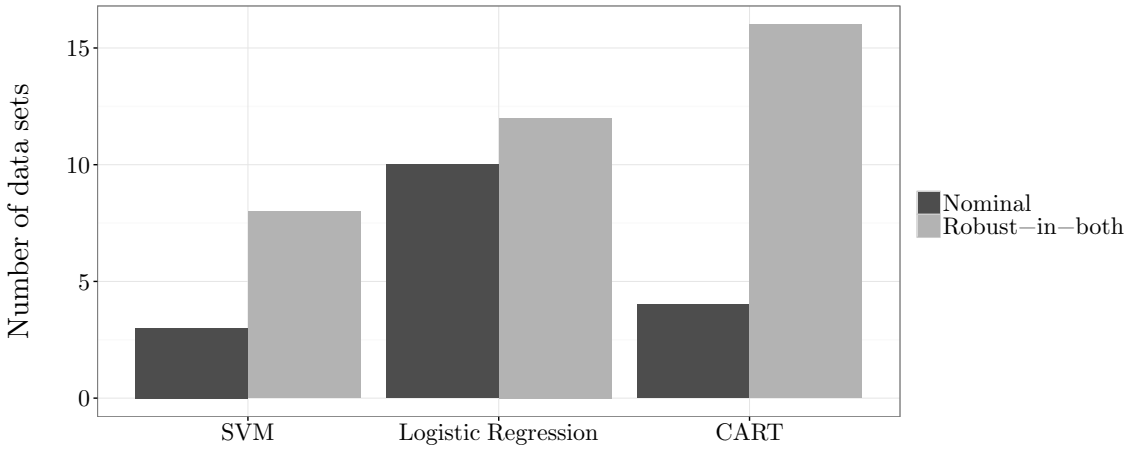


Figure 3-4: Comparison of the number of data sets for which the nominal and robust-in-both approaches for each method give the highest out-of-sample accuracy.

of the six methods yield the unique highest out-of-sample accuracy for certain data sets, which indicates that each type of classifier is able to exploit different aspects of the data set in their own ways to potentially lead to higher quality solutions. In all cases, the robust counterpart produced the highest number of uniquely optimal solutions.

3.8.4 Predicting the Effectiveness of Robust Classification

Thus far, we have demonstrated the strength of robust methods compared to their nominal counterparts over the set of 75 problems from the UCI Machine Learning Repository. For machine learning practitioners, we would also like to provide guidance about when it is worthwhile to use robust classification methods in practical applications. In this section, we consider the problem of predicting whether or not a robust classifier is likely to improve out-of-sample accuracy relative to the nominal method, using only the dimension of the training data and the accuracy of the nominal method on these data. Note that we consider in-sample nominal accuracy because this is an attribute of the training problem, and therefore is available at the validation stage when selecting the final classification method.

First we consider the influence of nominal in-sample accuracy in isolation. Table 3.5 shows the improvement in out-of-sample accuracy of robust-in-both methods

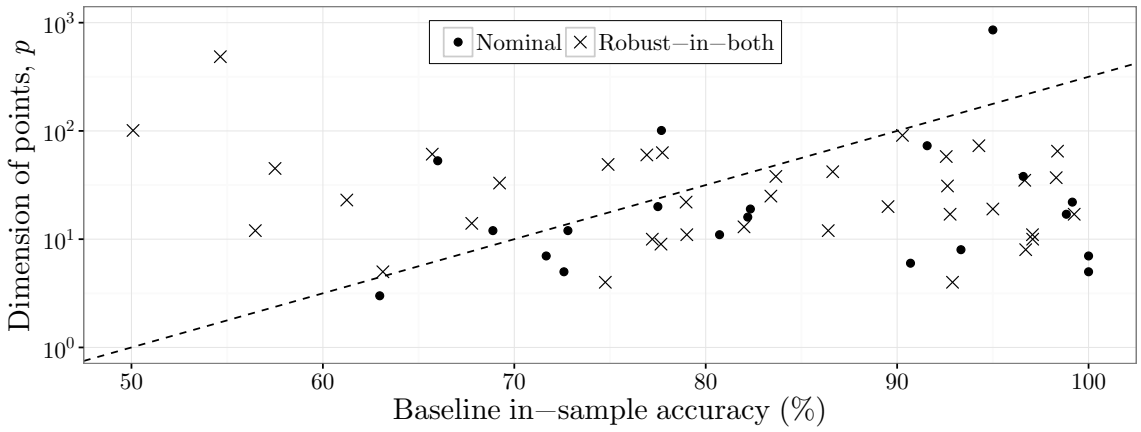
Table 3.5: Improvement due to robustness by baseline in-sample accuracy, comparing the baseline method to the corresponding robust-in-both classifier.

Nominal Method	Nominal Accuracy	Wins	Losses	Ties	Robust Improvement
SVM	0–60%	6	0	0	$10.7 \pm 5.6\%$
	60–70%	5	3	1	$2.2 \pm 1.9\%$
	70–80%	8	5	3	$0.9 \pm 1.1\%$
	80–90%	6	3	1	$0.3 \pm 0.5\%$
	90–100%	14	9	11	$0.0 \pm 0.4\%$
Logistic Regression	0–60%	2	1	0	$7.7 \pm 5.2\%$
	60–70%	10	0	0	$4.5 \pm 1.2\%$
	70–80%	8	7	0	$1.2 \pm 1.0\%$
	80–90%	4	5	1	$1.1 \pm 0.9\%$
	90–100%	16	7	14	$0.2 \pm 0.1\%$
CART	0–60%	1	0	2	$0.7 \pm 0.7\%$
	60–70%	1	0	0	$2.4 \pm -\%$
	70–80%	7	8	2	$0.1 \pm 0.9\%$
	80–90%	7	8	0	$-0.3 \pm 0.9\%$
	90–100%	17	8	14	$-0.1 \pm 0.6\%$

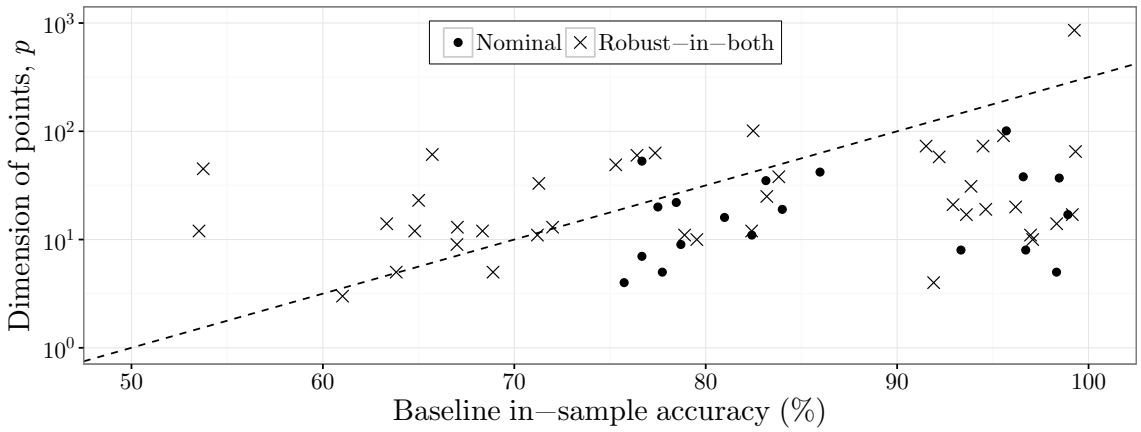
over their nominal counterparts for different ranges of nominal in-sample accuracy. We define the *robust improvement* as the absolute difference in out-of-sample accuracy between the methods, that is the accuracy of the robust-in-both method less the accuracy of the nominal method. For instance, if the robust-in-both and nominal methods had accuracies of 84.7% and 81.3%, respectively, the robust improvement would be +3.4%.

The most significant result is for data sets where nominal SVM has in-sample accuracy below 60%. For these 6 problems, robust-in-both SVM improves upon the out-of-sample accuracy in every instance, and yields an average robust improvement of 10.7%. For logistic regression and SVM, we see that as the nominal accuracy increases, both the proportion of robust-in-both wins and the robust improvement in accuracy decrease. For CART, the robust improvement is largely independent of the nominal accuracy, although the win proportion is higher for problems with nominal accuracy in the range of 90% to 100%. This suggests that nominal in-sample accuracy by itself is not a strong predictor of robust effectiveness for CART methods. However, note that there are only four data sets with a nominal CART accuracy below 70%, the region where the other robust methods are strongest.

(a) Nominal SVM vs. robust-in-both SVM



(b) Nominal logistic regression vs. robust-in-both logistic regression



(c) CART vs. robust-in-both Optimal Decision Trees

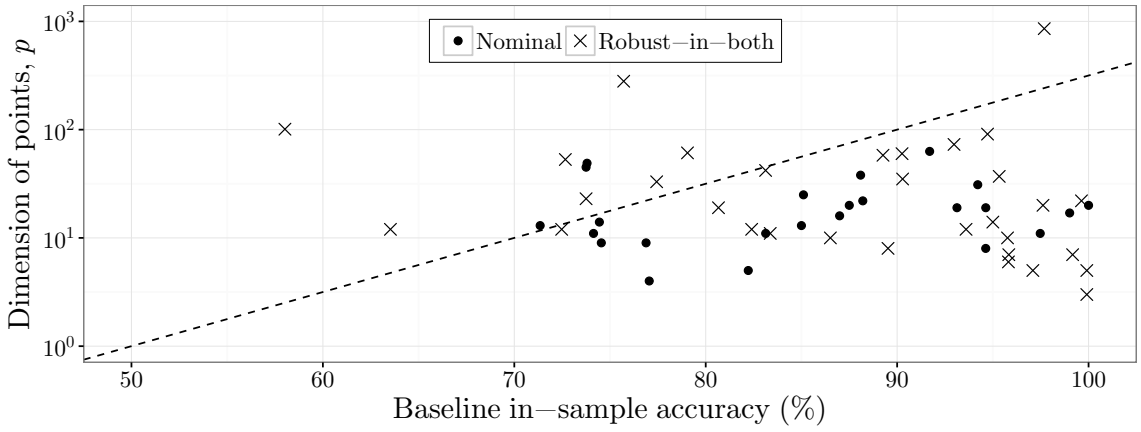


Figure 3-5: Plots of winning method (nominal vs. robust-in-both) by the baseline in-sample accuracy and dimension of points in each data set. The dashed line divides each plot into two regions with different levels of robustness gain. Nominal and robust-in-both wins are indicated by $\bullet$ and $\times$, respectively.

Table 3.6: Improvement due to robustness by baseline in-sample accuracy and dimension of points, comparing the nominal method to the corresponding robust-in-both classifier.

Baseline Method	Region	Wins	Losses	Ties	Robust Improvement
Nominal SVM	Above	14	4	3	$5.3 \pm 1.9\%$
	Below	25	16	13	$-0.2 \pm 0.3\%$
Nominal Logistic Regression	Above	17	2	1	$4.0 \pm 1.0\%$
	Below	23	18	14	$0.4 \pm 0.3\%$
Nominal Optimal Decision Trees	Above	7	3	4	$1.4 \pm 0.8\%$
	Below	17	25	19	$-0.7 \pm 0.4\%$
CART	Above	9	3	2	$1.3 \pm 0.9\%$
	Below	24	21	16	$-0.3 \pm 0.5\%$

Region Above refers to the top-left sections in Figure 5 (high data dimension, low baseline accuracy); Region Below refers to the bottom-right sections in Figure 5 (low data dimension, high baseline accuracy).

Next, we consider the combined influence of nominal in-sample accuracy and dimension of data points on the robust improvement. Figure 3-5 plots the winning method against these two attributes of the training problem. We have constructed a dividing line which is identical on all three plots that partitions the points into two regions. This line follows the equation $\log_{10}(p) = 0.05a - 2.5$, where a is the in-sample accuracy of the nominal method on the data set, p is the dimension of the data set, and the coefficients 0.05 and 2.5 were selected manually. In Table 3.6 we present a breakdown of the relative performance of the nominal and robust-in-both methods in the two regions. For all three classifiers, robust methods beat nominal methods for a majority of data sets in the region of lower nominal accuracy and high dimensionality (above the dividing line). In this region, we see significant average improvements in out-of-sample accuracy of 5.3% for SVM, 4.0% for logistic regression, and 1.3% for CART. Below the dividing line, we observe that robust methods are still competitive with nominal methods, with neither offering a significant advantage.

We also include in Table 3.6 a comparison of robust-in-both Optimal Decision Trees to nominal Optimal Decision Trees. Previously, we have only considered the performance relative to CART in order to provide a strong benchmark against the state-of-the-art methods, but it is also insightful to directly compare the robust for-

mulation to its nominal counterpart. Below the dividing line, the robust-in-both approach is not as strong compared to the Optimal Decision Trees as it is compared to CART. This can be attributed to the fact that the Optimal Decision Trees are a stronger classification method than CART, and thus provide a stronger nominal baseline. However, we see that above the line, the relative improvement of robust-in-both Optimal Decision Trees over Optimal Decision Trees is very similar to their improvement over CART, with an average improvement in out-of-sample accuracy of 1.4%. This therefore shows that the dividing line is a strong predictor for when robust methods perform strongest relative to nominal methods, even in the presence of a significantly stronger nominal method.

It seems natural that the data dimension and nominal accuracy are likely indicative of the problem difficulty. This implies that robust methods are most beneficial for harder problems. We also expect robust methods to perform strongest on noisy data. Together, this offers evidence that problem difficulty and data uncertainty are correlated, a result that is consistent with intuition. Based on the dividing line used earlier, we present the following decision rule to address the task of predicting the effectiveness of robust methods over nominal:

$$\log_{10}(p) \geq 0.05a - 2.5, \tag{3.31}$$

where p is the dimension of the data points, and a is the nominal in-sample accuracy. If this relationship is satisfied, the data set falls into the "Above" region of Table 3.6, and therefore the robust classification methods are highly likely to offer significant accuracy improvements over their nominal counterparts.

This demonstrates that we can predict with high-accuracy a significant improvement in out-of-sample accuracy when using robust methods for classification problems with high-dimensional data and low nominal accuracy. This has large practical importance for machine learning; given a real-world classification problem, (3.31) gives a simple but strong recommendation for when to use robust classification in place of nominal SVM, logistic regression, or CART.

3.8.5 Comparison with Regularized Methods

To demonstrate the added value of our principled framework for modeling data uncertainty with robust optimization, we compare the robust classification methods to other popular methods that exhibit robust properties indirectly.

Table 3.7 Out-of-sample accuracy averaged across five seeds for each method using both regularized and robust-in-both methods on all data sets.

Data Set Information			SVM		Logistic Regression	
UCI Data Set Name	n	p	Regularized	Robust	Regularized	Robust
acute-inflammations-1	120	7	**1.0000**	0.9083	**1.0000**	**1.0000**
acute-inflammations-2	120	7	**1.0000**	**1.0000**	**1.0000**	**1.0000**
arrhythmia	68	280	0.6154	**0.6308**	**0.7538**	0.6923
balance-scale	625	5	**0.9200**	**0.9200**	**0.9200**	**0.9200**
banknote-authentication	1372	5	0.9869	**0.9912**	0.9855	**0.9905**
blood-transfusion	748	5	**0.7638**	0.7638	0.7664	**0.7812**
breast-cancer	683	10	**0.9679**	0.9559	**0.9664**	0.9574
breast-cancer-diagnostic	569	31	**0.9719**	0.9614	**0.9684**	0.9684
breast-cancer-prognostic	194	33	**0.7692**	0.7179	**0.7692**	0.7590
car-evaluation	1728	16	**0.7977**	0.7826	**0.7936**	0.7925
chess-king-rook-vs-king	28056	35	**0.9004**	**0.9004**	**0.9004**	**0.9004**
chess-king-rook-vs-king-pawn	3196	38	**0.9743**	0.9687	**0.9756**	0.9743
climate-model-crashes	540	19	**0.9611**	0.9574	**0.9556**	0.9537
cnae-9	1080	857	**0.9769**	0.9481	0.9713	**0.9824**
congressional-voting-records	232	17	0.9787	**0.9826**	0.9574	**0.9826**
connectionist-bench	990	11	0.9737	**0.9768**	0.9707	**0.9768**
connectionist-bench-sonar	208	61	0.7268	**0.7561**	0.7073	**0.7659**
contraceptive-method-choice	1473	12	**0.6800**	0.6755	**0.6814**	0.6776
credit-approval	653	38	**0.8626**	0.8585	**0.8733**	0.8631
cylinder-bands	277	485	**0.7200**	0.6691	0.6182	**0.6727**
dermatology	358	35	**0.9915**	0.9803	**1.0000**	**1.0000**
echocardiogram	61	7	**0.7000**	0.6833	0.6667	**0.7333**
ecoli	336	8	**0.9791**	0.9582	**0.9731**	0.9582
fertility	100	13	0.8500	**0.9000**	0.8400	**0.9000**
flags	194	60	**0.8872**	0.8205	**0.8615**	0.8564
glass-identification	214	10	**0.7302**	0.7256	**0.7395**	0.7349
haberman-survival	306	4	0.7279	**0.7344**	0.7180	**0.7311**
hayes-roth	132	5	**0.8519**	0.6692	**0.8074**	0.7923
heart-disease-cleveland	297	19	**0.8305**	0.8203	**0.8441**	0.8305
hepatitis	80	20	**0.8375**	0.8125	0.8125	**0.8250**
hill-valley	606	101	0.8364	**0.9620**	**0.9884**	0.9636

For each data set, the best result (or both in the case of a tie) for each method is indicated in bold.

Table 3.7 (Cont.) Out-of-sample accuracy averaged across five seeds for each method using both regularized and robust-in-both methods on all data sets.

Data Set Information			SVM		Logistic Regression	
UCI Data Set Name	n	p	Regularized	Robust	Regularized	Robust
hill-valley-noise	606	101	0.8132	**0.8512**	0.8678	**0.8876**
image-segmentation	210	20	**0.9905**	0.9476	**0.9810**	0.9762
indian-liver-patient	579	11	**0.7155**	**0.7155**	**0.7224**	0.7224
ionosphere	351	35	**0.8743**	**0.8743**	**0.8943**	0.8714
iris	150	5	**1.0000**	0.9800	**1.0000**	**1.0000**
letter-recognition	20000	17	0.9916	**0.9923**	0.9904	**0.9908**
libras-movement	360	91	**0.9694**	**0.9694**	0.9583	**0.9639**
magic-gamma-telescope	19020	11	0.7848	**0.7924**	0.7862	**0.7919**
mammographic-mass	830	11	**0.8120**	0.8060	**0.8301**	0.8217
monks-problems-1	124	12	0.6960	**0.8000**	0.6560	**0.7920**
monks-problems-2	169	12	0.5824	**0.6176**	0.5882	**0.6235**
monks-problems-3	122	12	**0.9360**	0.9333	**0.9360**	0.9250
mushroom	5644	77	**1.0000**	**1.0000**	**1.0000**	**1.0000**
nursery	12960	20	**1.0000**	**1.0000**	**1.0000**	**1.0000**
optical-recognition	3823	65	0.9956	**0.9966**	0.9958	**0.9971**
ozone-level-detection-eight	1847	73	**0.9355**	0.9295	**0.9366**	0.9317
ozone-level-detection-one	1848	73	**0.9702**	**0.9702**	0.9675	**0.9702**
parkinsons	195	22	**0.8872**	0.8615	**0.8462**	0.8205
pen-based-recognition	7494	17	**0.9893**	0.9891	0.9896	**0.9897**
pima-indians-diabetes	768	9	0.7647	**0.7791**	0.7660	**0.7752**
planning-relax	182	13	0.7027	**0.7222**	**0.7027**	0.7000
poker-hand	25010	11	0.5018	**0.5023**	**0.5005**	0.5000
post-operative-patient	87	14	**0.7059**	**0.7059**	**0.7059**	0.6588
qsar-biodegradation	1055	42	0.8692	**0.8758**	0.8578	**0.8682**
seeds	210	8	0.9333	**0.9429**	**0.9619**	0.9476
seismic-bumps	2584	21	**0.9342**	**0.9342**	**0.9335**	0.9327
skin-segmentation	245057	4	0.9326	**0.9366**	0.9187	**0.9345**
soybean-large	266	63	**0.9094**	0.8717	**0.9170**	0.8868
soybean-small	47	38	**1.0000**	**1.0000**	**1.0000**	**1.0000**
spambase	4601	58	**0.9287**	0.9265	0.9241	**0.9246**
spect-heart	80	23	0.6375	**0.7125**	0.6750	**0.7625**
spectf-heart	80	45	0.6375	**0.6500**	0.6750	**0.7625**
statlog-project-german-credit	1000	49	**0.7420**	0.7400	0.7350	**0.7400**
statlog-project-landsat-satellite	4435	37	**0.9867**	0.9820	**0.9851**	0.9824
teaching-assistant-evaluation	151	53	**0.7200**	0.6733	**0.8067**	0.7067
thoracic-surgery	470	25	**0.8511**	**0.8511**	**0.8532**	0.8468
thyroid-disease-ann-thyroid	3772	22	0.9905	**0.9915**	0.9910	**0.9934**
thyroid-disease-new-thyroid	215	6	0.8837	**0.8884**	0.8977	**0.8977**

For each data set, the best result (or both in the case of a tie) for each method is indicated in bold.

Table 3.7 (Cont.) Out-of-sample accuracy averaged across five seeds for each method using both regularized and robust-in-both methods on all data sets.

Data Set Information			SVM		Logistic Regression	
UCI Data Set Name	n	p	Regularized	Robust	Regularized	Robust
tic-tac-toe-endgame	958	19	**0.9812**	0.9801	**0.9801**	0.9801
wall-following-robot-2	5456	3	**0.6440**	0.5553	0.6537	**0.6609**
wall-following-robot-24	5456	5	0.6436	**0.6561**	**0.6565**	0.6563
wine	178	14	**0.9829**	0.9657	0.9886	**0.9943**
yeast	1484	9	0.6869	**0.6902**	0.6842	**0.6929**
zoo	101	17	**1.0000**	**1.0000**	**1.0000**	**1.0000**

For each data set, the best result (or both in the case of a tie) for each method is indicated in bold.

First, we compare our feature-robust SVM to ℓ_1-regularized SVM, which is equivalent to classical SVM except for the ℓ_1 norm regularizer term. This is a feature-robust method under a different uncertainty set (see Section 3.4.2). We implemented Problem (3.3) in JuMP and solved this problem with Gurobi. Experimentally, feature-robust SVM and ℓ_1-regularized SVM produce comparable classifiers; across all 75 data sets analyzed, the average difference in out-of-sample accuracy between these two methods was $0.2 \pm 0.4\%$. This therefore gives evidence that our proposed uncertainty set for feature-robustness is an equally strong model of the uncertainty in the features of the data.

Next, to benchmark robust-in-both methods against regularized methods, we compare robust-in-both SVM against ℓ_1-regularized SVM, and robust-in-both logistic regression against ℓ_1-regularized logistic regression (which uses an ad-hoc method for introducing robustness). For ℓ_1-regularized logistic regression, we implemented Problem (3.5) with $q = 1$ in JuMP and solved this problem with Ipopt. We present the accuracy results for this comparison in Table 3.7.

In Table 3.8, we present the relative performance of robust-in-both and regularized methods broken down into the same two regions as defined in Section 3.8.4. As before, the regions are determined by the in-sample accuracy of the non-robust method and the data dimension. We see that for both SVM and logistic regression, robust methods still offer improved accuracy over regularized methods for a majority of data sets in the

Table 3.8: Improvement due to robustness by baseline in-sample accuracy and dimension of points, comparing the regularized method to the corresponding robust-in-both classifier.

Baseline Method	Region	Wins	Losses	Ties	Robust Improvement
Regularized SVM	Above	8	6	1	$0.5 \pm 1.1\%$
	Below	18	29	13	$-0.7 \pm 0.5\%$
Regularized Logistic Regression	Above	8	5	0	$1.9 \pm 1.6\%$
	Below	24	28	10	$0.1 \pm 0.3\%$

Region Above refers to the top-left sections in Figure 5 (high data dimension, low baseline accuracy); Region Below refers to the bottom-right sections in Figure 5 (low data dimension, high baseline accuracy).

region of lower nominal accuracy and high dimensionality (above the dividing line). In this region, we see average improvements in out-of-sample accuracy of 0.5% over regularized SVM and 1.9% over regularized logistic regression. Below the dividing line, we observe that robust methods are still competitive with nominal methods, although regularized SVM outperforms robust SVM by 0.7% in this region. If we consider alternate norms and compare robust SVM and logistic regression against ℓ_2-regularized methods instead, we obtain similar results.

These results demonstrate that classifiers do benefit from a principled approach to robustness evidenced in real-world data, even when compared to regularized methods that are stronger than nominal ones. In all cases, we observe that our robust methods perform best on classification problems which satisfy the decision rule given by equation (3.31).

3.8.6 Computational Tractability and Speed

Table 3.9 shows the complexity of each nominal method and its robust counterparts. Under all three classifiers, the feature-robustness does not change the nature of the optimization problem complexity. Logistic regression changes from unconstrained convex optimization to constrained when label-robustness is introduced. Label-robust SVM introduced integer-valued variables and therefore becomes a mixed-integer optimization problem. For Decision Trees, since the nominal formulation is mixed-integer optimization formulation, label robustness does not change the nature of the problem.

Table 3.9: Problem complexity of nominal and robust classification methods.

Method	Nominal	Feature-robust	Label-robust	Robust-in-both
SVM	LO	LO	MIO	MIO
Logistic Regression	Unconstr. CO	Unconstr. CO	Constr. CO	Constr. CO
Decision Trees	MIO	MIO	MIO	MIO

Robustness-in-both takes the maximum complexity between feature-robust and label-robust formulations; in this case, the complexity is equal to that of the label-robust in all three classifiers.

In order to provide empirical measures of the complexity of each method, we also compare the total time required to solve a problem instance for each method with or without robustness across a selection of UCI data sets. These sets are chosen to be representative of the various dimensions and separability among all data sets. For the robust methods, a typical choice of $\rho = 0.01$, $\Gamma = 10\%$ is used. The problems were solved on a machine with a 16-core, Intel Xeon E5-2687W (3.1 GHz) Processor and 128 GB RAM and the total solver time taken to solve each problem instance to optimality was recorded. All tests were limited to a single thread for consistency. If the problem was not solved to optimality within an hour, the solve was terminated. In this case, we report the time taken to find the solution that was best under the hour time limit. In particular for robust counterparts of CART, strong heuristics give very good solutions almost instantly, and sometimes these solutions are not further improved after an hour. In a real-world application of these methods, the time taken to find the solution is the more important measure than the time taken to prove the solution optimal; therefore time to finding solution is used.

The results for selected data sets are presented in Table 3.10. In general, the nominal and feature-robust classifiers require solver time of around the same order of magnitude. Label robustness generally slows down computation by 1–2 orders of magnitude; in particular, since label-robustness for SVM changes the problem from a linear optimization problem to a mixed-integer optimization problem, the computational time is considerably longer. The robust-in-both classifier tends to exhibit similar solution times to the label-robust method.

Table 3.10: Solver time for selected UCI data sets in seconds for $\rho = 0.01$ and $\Gamma = 10\%$.

Method	Type of Robustness	UCI Data Set (number of points, dimension)					
		hayes-roth (132, 4)	bank. auth. (1372, 4)	nursery (12960, 19)	skin seg. (245057, 3)	flags (194, 59)	cnae-9 (1080, 856)
SVM	Nominal	0.00	0.02	0.05	454.38	0.01	0.02
	Feature	0.00	0.02	0.36	553.94	0.01	0.32
	Label	0.23	4.50	58.58	695.06*	0.37	2.41
	Both	0.24	4.77	91.70	695.06*	0.60	15.81
Logistic regression	Nominal	0.00	0.05	0.02	0.03	0.03	0.41
	Feature	0.00	0.08	0.03	0.16	0.16	113.24
	Label	0.03	0.24	4.70	56.33	0.06	0.52
	Both	0.03	0.25	5.45	71.12	0.06	0.51
Decision trees	Nominal	0.02	0.02	0.18	1.44	0.02	0.65
	Feature	0.04	0.02*	1.06	1.46*	0.64	0.65*
	Label	3.39	45.00*	0.18*	1.47*	3.01	183.43
	Both	0.05	—[a]	0.18*	1.48*	2.39	146.01

* Not solved to optimality within the time limit. The time reported is instead the time taken to find the solution that is best at termination.
[a] The robust-in-both optimal decision tree problem is infeasible for this particular choice of ρ/Γ.

3.8.7 The Price of Robustness

Introducing robustness in classifiers generates solutions that may be suboptimal under the nominal data, but are likely to remain feasible or close to optimal when the data change [19]. We can evaluate this trade-off for the robust classifiers by comparing the out-of-sample accuracies, as evaluating the model accuracy on the unobserved testing data can be thought of as a way of exposing the solution to perturbations in the training data.

The empirical findings show that robustness improves prediction accuracy in many real-world data sets across all three classifications methods. In each classifier family individually, feature-robust, label-robust, and robust-in-both generally have higher winning counts compared to their nominal counterpart. When comparing all three nominal methods and their robust versions together, robustness continues to perform well in the majority of data sets, particularly in subsets of data sets that are more difficult to classify. Overall, robust methods offer quality solutions that nominal ones cannot achieve.

Another practical aspect on the price of robustness is the computational time requirement. In most cases, the computational time for robust methods is on the same

order of magnitude as their respective nominal ones, suggesting that robustification does not incur a significant burden on speed. It should also be noted that as mixed integer optimization problems, label-robust SVM and CART can easily be limited by computational constraints. Several problems we considered were not solved to optimality, rather stopped after a smaller time limit to get a strong, yet suboptimal, solution. Allowing for longer time limits in these cases has the potential for further improving the accuracy.

3.9 Conclusions

In this chapter, we consider three major classification methods under a modern Robust Optimization perspective: SVM, logistic regression, and CART. For each classifier, we address uncertainties in features, labels, and both simultaneously in a principled manner by constructing appropriate uncertainty sets and deriving robust counterparts in the same way for all methods. We also discuss the implementation and practical solvability for each method with robustness.

Synthetic experiments demonstrate that our methods derived by taking a principled approach to robust classification may improve greatly upon existing classification methods. In the synthetic study, we show that robust-in-both SVM and logistic regression outperform both nominal and regularized methods and produce classifiers closer to the underlying truth, especially in the worst case scenarios. In particular, the 90^{th} percentile out-of-sample errors for our methods are significantly lower than the 90^{th} percentile out-of-sample errors for the benchmark methods. Because regularized SVM can be cast as a feature-robust optimization problem for a particular uncertainty set, this shows that the choice of uncertainty set may be critical. For the simple synthetic problems considered here, the robust methods derived using label uncertainty sets perform best.

To evaluate the value of adding robustness in practice, we performed computational experiments on a large sample of data sets from the UCI Machine Learning Repository, comparing nominal, regularized, and robust methods for each of the three

classifiers. We find that robust solutions provide higher out-of-sample accuracy for many data sets, and the large majority of classifiers which strictly outperformed all other methods were robust. In particular, we identify that high-dimensional and hard-to-separate problems benefit most from our principled approach to robustness. The findings suggest that we can predict how much value robustness will add to a data set given only the accuracy of a classical method and dimension of the data set features. This allows us to offer guidance as to when robust classification methods can deliver significant improvements in practical settings.

3.10 Equivalence with Classical Support Vector Machines

The feature-robust counterpart presented in Theorem 1 is similar to the classical SVM problem (3.2). Making the substitution $\tilde{\xi}_i = \xi_i - \rho\|\mathbf{w}\|_{q^*}$ in Problem (3.11), we obtain

$$\begin{aligned} \min_{\mathbf{w},b} \; & n\rho\|\mathbf{w}\|_{q^*} + \sum_{i=1}^{n} \tilde{\xi}_i \\ \text{s.t. } & y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \tilde{\xi}_i \qquad i = 1, \ldots, n, \\ & \tilde{\xi}_i \geq -\rho\|\mathbf{w}\|_{q^*} \qquad\qquad i = 1, \ldots, n. \end{aligned} \tag{3.32}$$

Comparing Problem (3.32) to the classical SVM formulation (3.2), we observe that adding feature robustness or regularization to the hinge loss classifier lead to nearly identical optimization problems. Depending upon the choice of uncertainty set and the selection of the regularizing term, this equivalence may be exact. Under the assumption that the training data are non-separable, [45] has shown that the robust optimization problem

$$\begin{aligned} \min_{\mathbf{w},b} \; & \max_{\Delta\mathbf{X}\in\tilde{\mathcal{U}}_x} \sum_{i=1}^{n} \xi_i \\ & \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i \qquad i = 1, \ldots, n, \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad i = 1, \ldots, n, \end{aligned} \tag{3.33}$$

is exactly equivalent to the problem

$$\begin{aligned} \min_{\mathbf{w},b} \ & \rho\|\mathbf{w}\|_{q^*} + \sum_{i=1}^{n} \xi_i \\ \text{s.t. } & y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i \qquad i = 1, \ldots, n, \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad i = 1, \ldots, n, \end{aligned} \tag{3.34}$$

where

$$\tilde{\mathcal{U}}_x = \left\{ \mathbf{\Delta X} \in \mathbb{R}^{n \times p} \;\middle|\; \sum_{i=1}^{n} \|\mathbf{\Delta} x_i\|_q \leq \rho \right\}.$$

It follows that (3.34) is equivalent to the classical SVM problem (3.2) for the choice of $q^* = 2$, or the ℓ_1-regularized SVM problem (3.3) for the choice of $q^* = \infty$. This implies that the classical and regularized SVM problems are indeed robust formulations of the nominal hinge loss classifier under specific choices of uncertainty set.

Chapter 4

Personalized Diabetes Management Using Electronic Medical Records

This work appeared in Diabetes Care, with co-authors Dimitris Bertsimas, Nathan Kallus, and Alexander Weinstein [11].

Current clinical guidelines for managing type 2 diabetes do not differentiate based on patient-specific factors. In this chapter, we present a data-driven algorithm for personalized diabetes management that improves health outcomes relative to the standard of care.

We modeled outcomes under 13 pharmacological therapies based on electronic medical records from 1999 to 2014 for 10,806 patients with type 2 diabetes from Boston Medical Center. For each patient visit, we analyzed the range of outcomes under alternative care using a k-nearest neighbor approach. The neighbors were chosen to maximize similarity on individual patient characteristics and medical history that were most predictive of health outcomes. The recommendation algorithm prescribes the regimen with best predicted outcome if the expected improvement from switching regimens exceeds a threshold. We evaluated the effect of recommendations on matched patient outcomes from unseen data.

Among the 48,140 patient visits in the test set, the algorithm's recommendation mirrored the observed standard of care in 68.2% of visits. For patient visits in which the algorithmic recommendation differed from the standard of care, the mean post-treatment glycated hemoglobin A1c (HbA1c) under the algorithm was lower than standard of care by $0.44 \pm 0.03\%$ ($p \ll 0.001$), from 8.37% under the standard of care to 7.93% under our algorithm.

A personalized approach to diabetes management yielded substantial improvements in HbA1c outcomes relative to the standard of care. Our prototyped dashboard visualizing the recommendation algorithm can be used by providers to inform diabetes care and improve outcomes.

4.1 Introduction

Diabetes is a chronic condition affecting almost 10% of the US population [75]. Individuals with diabetes experience abnormally high blood glucose levels, which can lead to severe complications such as heart disease, stroke, and kidney failure. The most common form of diabetes is type 2 diabetes, which constitutes 90-95% of all diabetes cases in the US [35]. The disease is typically managed through healthy eating, physical activity, oral medication, and/or insulin injections. While there are evidence-based clinical guidelines for glycemic control [80], how to choose among pharmacological therapies to maximize effectiveness for a given patient is not well understood. There has been growing interest in using clinical evidence to understand the effects of treatments in different type 2 diabetes populations. In a joint statement from 2012, the American Diabetes Association and the European Association for the Study of Diabetes highlighted the need for a patient-centered approach to diabetes management [59]. The need for an individualized approach is especially pressing given the variety of disease symptoms, comorbid conditions, pharmacological treatments, individual treatment histories, and other individual characteristics that may inform treatment [87].

Evidence suggests that the response to blood glucose regulation agents can differ

among population subgroups. A post-hoc secondary analysis found that African-American pre-diabetic adults responded better to metformin than Caucasian pre-diabetic adults [100]. Another study recommended less aggressive treatments for older patients, as they were more likely to experience severe consequences from hypoglycemia [60]. These studies each provide valuable insights with respect to a single subgroup or treatment, but do not offer a decision rule for the general population that providers can easily apply in practice.

Tailoring glycemic management for specific subpopulations can be critical. Among patients with chronic kidney disease, contraindication to metformin needs to be taken into consideration when prescribing medication [68]. Separate glycated hemoglobin (HbA1c) goals may be needed for subgroups or individuals differentiated by age, comorbidities, and other clinical characteristics [87]. A personalized treatment recommendation using a quantitative approach could readily incorporate different glycemic targets and contraindications, and thus allow for more systematic management of subgroups.

We provide an algorithm that generates a personalized type 2 diabetes treatment recommendation for any given patient based on evidence from historical outcomes of similar patients drawn from an electronic medical records (EMR) database. EMR analysis allows for pinpoint comparisons of effectiveness because of the abundance of clinical evidence from multiple treatment options administered to a diverse population over long-term patient clinical histories. EMR data combines the large sample sizes found in some insurance claims databases with the depth of longitudinal clinical evidence typically found in clinical trials. One caveat is that EMR data are not controlled via randomization.

Our methodological approach applies machine learning techniques and causal inference to make personalized recommendations based on comparative effectiveness among subpopulations in the EMR database. Machine learning techniques have been increasingly adopted in health care, along with many other fields [61, 10, 17]. Our novel approach leverages the power of analytics and abundant data in the EMR system to improve quality of care.

The recommendations are personalized by patient characteristics, including age, sex, race, BMI, treatment history, and diabetes progression. We evaluate the effectiveness of the personalized treatment recommendations against the current standard of care by estimating patients' counterfactual outcomes from historical outcomes of similar patients in the EMR database. We develop a prototype clinical support dashboard that provides evidence for the algorithm's recommendations and could guide providers in caring for type 2 diabetes patients in a personalized manner.

4.2 Research Design and Methods

4.2.1 Analytic overview

We modeled outcomes for patients with type 2 diabetes based on EMR data. We divided each patient's medical history into distinct lines of therapy, each characterized by a particular drug monotherapy or combination therapy. Within each line of therapy, we considered patient visits occurring every 100 days. At each visit, the provider decides whether to proceed with the patient's current line of therapy or to recommend an alternative regimen. We developed a non-parametric prescriptive algorithm that provides personalized treatment recommendations. For each patient visit, we used k-nearest neighbor (kNN) regression [37] to predict the potential HbA1c outcome under each treatment alternative. The nearest neighbors were chosen to control for confounding that may be present in non-randomized data [81] and to maximize similarity on the patient characteristics that were most predictive of outcomes. The algorithm then prescribed the regimen with best predicted outcome, provided the predicted improvement relative to the patient's current regimen exceeded a confidence threshold. The outcome metric was the average HbA1c measurement 75 to 200 days after the visit date. The effect of the prescriptive algorithm was evaluated by comparing the expected HbA1c outcome under our recommended therapy to the observed outcome under the standard of care (ground-truth) therapy, according to a commonly used matching approach [58]. We conducted additional simulations to

ensure that the results were robust to training models on different datasets and using alternative predictive modeling techniques.

4.2.2 Data

Through a partnership with Boston Medical Center (BMC), an academic medical center in Boston, Massachusetts, we obtained EMR for over 1.1 million patients from 1999 to 2014. In this dataset, 10,806 patients met all of the following inclusion criteria:

- Were present in the system for an observation period of at least 1 year;
- Received a prescription for at least one blood glucose regulation agent, including insulin, metformin, sulfonylureas, or one of the other blood glucose regulation agents listed below, and had at least one medical record 100 days prior to the date of this prescription;
- Had at least three recorded laboratory measurements of HbA1c; and,
- Did not have a recorded diagnosis of type 1 diabetes, as defined by the presence of International Classification of Diseases (ICD-9) diagnosis code 250.x1 or 250.x3 combined with the absence of any subsequent prescriptions for oral blood glucose regulation agents. (If the patient received oral blood glucose regulation agents subsequent to one of these diagnosis codes, we assumed the diagnosis record was an error.)

For each patient, we had access to demographic data, including date of birth, sex, and race/ethnicity, and to all BMC EMR data, including a history of drug prescriptions and measurements of height, weight, BMI, and HbA1c, as well as creatinine levels (Table 4.1). Neither the size of the population nor the proportion with good glycemic control changed substantially over the course of the study.

4.2.3 Interpreting individual medical histories

We divided each patient's medical history into distinct lines of therapy, each characterized by a particular drug regimen (Figure 4-1). Within each line of therapy, we

Table 4.1: Demographics, medical history, and treatment history of patients ($N = 10,806$).

Feature	**Mean (SD)**
*Age (years)**	59.7 (13.6)
% Male	42.4%
% Black	58.5%
% Hispanic	15.1%
% White	16.6%
*BMI (kg/m^2)**	33.1 (8.1)
*HbA1c (%)**	7.9 (1.8)
% with good glycemic control, i.e. HbA1c $\leq 7.0\%$*	37.7%
Years since first treatment in EMR	3.52 (3.66)
Current prescription for metformin†	45.6%
Current prescription for insulin†	30.2%
Contraindicated to metformin‡	17.4%
Number of patients with first visit prior to 2007 (%)	6,175 (57.1%)

* Sample statistics are calculated across all patient visits. Individual patients with longer medical histories may be over-represented in the sample.
† Individuals may have a current prescription for both metformin and insulin.
‡ A patient was considered to be contraindicated to metformin when current serum level of creatinine was greater than 1.5 mg/dL.

considered patient visits occurring every 100 days, corresponding to the life cycle of a red blood cell [47]. These patient visits provided the basis for our definition of patient outcomes.

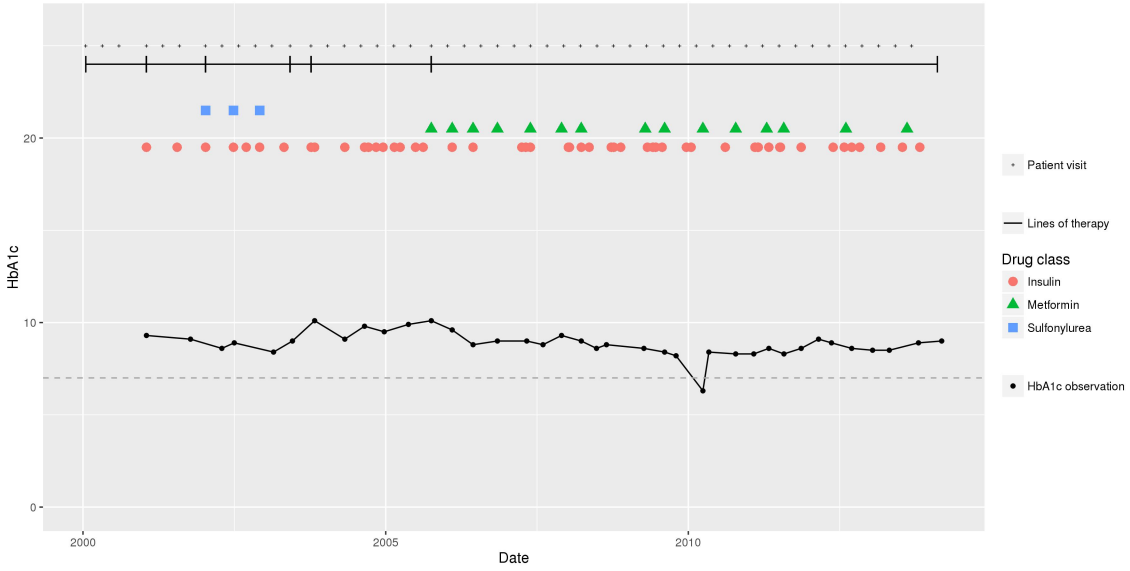


Figure 4-1: Treatment history for a sample patient.

The black points connected by black lines in the lower portion of the figure depict the HbA1c measurements over time for an example patient. The dotted horizontal black line shows a desired HbA1c threshold of 7%. The history of prescriptions this patient has received for each of three drugs, insulin, metformin, and sulfonylureas, is shown in red, green, and blue, respectively. The start and end of each line of treatment is indicated by a vertical tick mark. Finally, above the phase descriptions are small black dots indicating the date of each patient visit.

Lines of Therapy. We developed an algorithm to define precisely when each line of therapy ends and the next line begins according to when the combination of drugs prescribed to the patient changes in the EMR data. Each line of therapy was characterized by a unique drug regimen, defined to include all blood glucose regulation agents prescribed to the patient within the first 6 months after starting that line of therapy.

Regimens were defined as combinations of drugs from one or more drug classes. The drug classes we considered were metformin, insulin, and other blood glucose regulation agents; the other agents included sulfonylureas, thiazolidinediones, DPP-

4 inhibitors, meglitinides, alpha-glucosidase inhibitors, GLP-1 agonists, and other antihyperglycemic agents. If a sufficient number of HbA1c observations existed during a period in which no drugs were prescribed, we defined the patient's line of therapy as "NoRx." We considered thirteen possible regimen types (Table 4.2). A combination of drug classes was included as a regimen type if it was observed in a sufficient number of patient visits.

Table 4.2: Pharmacological regimens.

Observed standard of care regimen (Abbreviation)	**Number of patient visits**
No regimen prescribed, new patient (NEWPT)	5,449
No regimen prescribed, existing patient (NORX)	2,137
Metformin monotherapy (MET0)	9,649
Insulin monotherapy (INS0)	7,539
Other blood glucose regulation agent monotherapy (OTHER0)	4,671
Metformin combined with one other non-insulin agent (MET1)	6,959
Metformin combined with insulin (METINS0)	3,977
Insulin combined with one non-metformin oral agent (INS1)	2,139
Combination of two non-metformin, non-insulin agents (OTHER1)	1,047
Metformin combined with two other oral agents (MET2)	1,749
Metformin combined with insulin and one other agent (METINS1)	2,005
Insulin combined with two non-metformin agents (INS2)	249
All other multi-drug (3+) combinations (MULTI)	570
Total	48,140

Patient Visits. Within each line of therapy, we considered patient visits occurring every 100 days, beginning with the visit at which that regimen was initiated and continuing until no later than 80 days prior to the start of the subsequent regimen. There were 48,140 unique patient visits in our dataset (Table 4.2). At each visit, we defined a set of visit-specific patient characteristics, including the current line of therapy (i.e. therapy given during the 100 days immediately preceding the current visit) and recent HbA1c and BMI history. The outcome was measured as average HbA1c 75 to 200 days after the visit. This effect period was chosen to allow for a complete red blood cell life cycle to elapse before measuring the effect of a drug

therapy.

We defined the standard of care for each visit as the drug regimen which was administered. For 16.3% of visits, the provider prescribed an adjustment to the current line of therapy; in the other 83.7%, the provider's prescription was to continue the current regimen.

4.2.4 Prescriptive algorithm

Our novel prescriptive algorithm considers a menu of available treatment options, including the patient's current treatment; uses k-nearest neighbor regression models to predict potential outcomes under each option; rejects any non-current treatment option with predicted outcome above a pre-specified HbA1c threshold; and chooses the remaining option with best predicted outcome. The menu of options for a given patient could be determined by the provider, accounting for contraindications and other preferences, such as not using intensive control for elderly patients or patients with a history of severe hypoglycemia.

For the purposes of this analysis, the menu of options for each patient was chosen relative to the intensity and composition of the patient's current treatment regimen. Specifically, the algorithm considered only regimens that represented an incremental addition or subtraction of a drug, or substitution of a drug of comparable intensity; metformin and insulin were considered to be of the lowest and highest intensity, respectively. Patients with serum creatinine levels, greater than 1.5 mg/dL [68], a sign of kidney disease, were not offered metformin-based regimens. The menu options used in our analysis, differentiated by current treatment, are depicted in Figure 4-2; by definition, the algorithm never recommended metformin-based therapies for patients with the contraindication described above.

For each patient visit, the outcomes predicted by kNN under each treatment were compared. Our algorithm selected the treatment with the best predicted HbA1c outcome subject to the condition that this best predicted outcome improve upon the predicted outcome under the patient's current treatment by at least some threshold δ. We chose the optimal threshold value of 0.8% by testing the algorithm on a single test

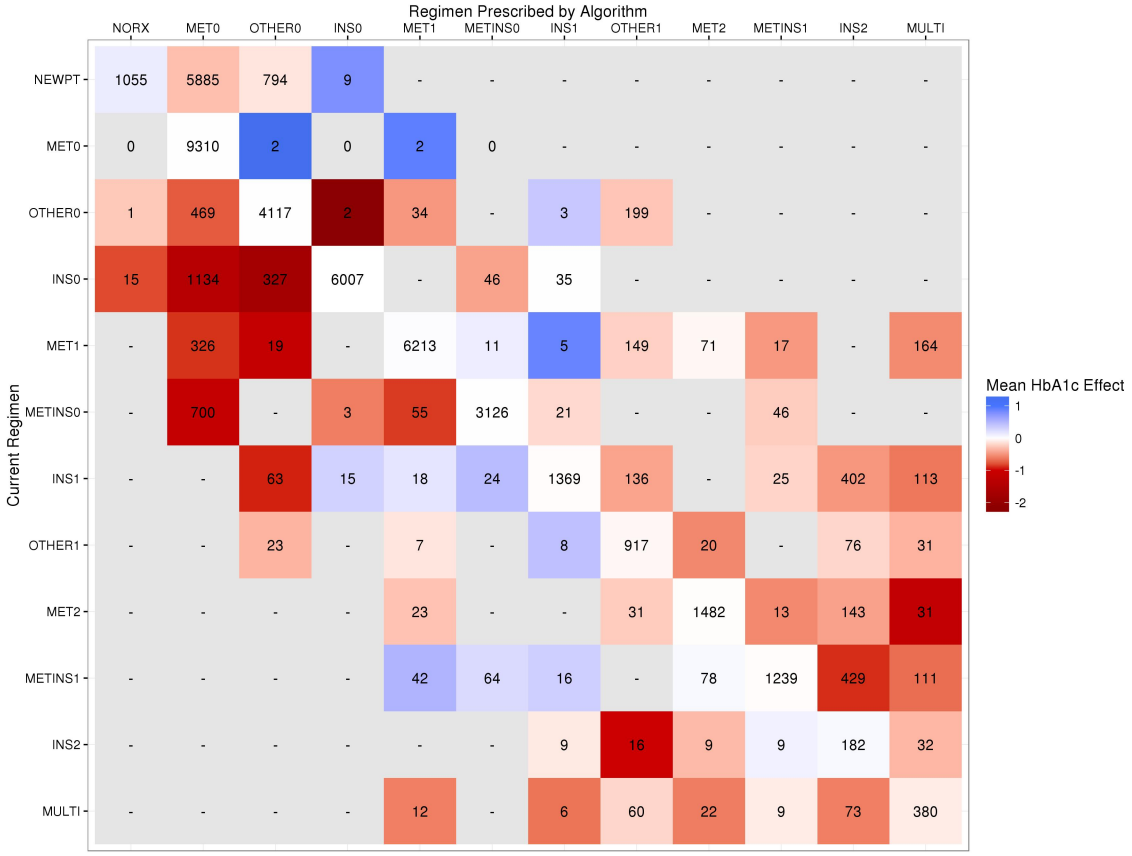


Figure 4-2: HbA1c benefit of prescriptive algorithm for patients switching regimens.

Each cell in the figure represents patients for whom the prescriptive algorithm recommended switching from the regimen on the vertical axis to the regimen on the horizontal axis. The color in each cell indicates the mean HbA1c benefit (%) of the prescriptive algorithm for patients in that cell, with red indicating benefits of the algorithm and blue indicating worsening relative to standard of care. Each cell is labeled with the number of patients who made that switch; cells labeled with a dash were not on the menu of options provided to patients currently on a given regimen. Patients with serum creatinine levels greater than 1.5 mg/dL were not considered for metformin-based regimens, and therefore are never assigned by the algorithm to columns with metformin-based regimens.

set, using values of δ ranging from 0% to 1.5%. Increasing the threshold δ causes the algorithm to recommend switching for fewer patients, but the mean benefit among those who switch increases. Above a certain threshold, the recommendation fits to noise in the training data and does not provide better mean benefits in the testing set. The optimal threshold balances these concerns.

kNN regression is a non-parametric, instance-based algorithm that makes predictions by averaging the outcomes for the subset of observations most similar to the target as defined by some distance metric [37]. To predict potential outcomes under each regimen, we used a kNN regression based on a treatment-specific weighted Euclidean distance across normalized patient and visit-specific factors. The weights were derived by training a separate ordinary least squares linear regression model for each treatment regimen and using the magnitudes of the regression coefficients (Figure 4-3). This weighted distance improves upon classical kNN by selecting neighbors based on the factors most predictive of HbA1c outcome, rather than weighting all factors equally.

We considered factors from the following categories: demographic information, medical history, and treatment history. Specifically, the demographic factors used in the model were age, sex, and race. The medical history factors were days since first diabetes diagnosis; the patient's average serum creatinine level in the previous year; the patient's past two HbA1c and most recent BMI observations up to and including the current visit; the patient's average, median, 25th percentile, and 75th percentile HbA1c and BMI in the 1000-day period up to and including the current visit; and the patient's frequency of HbA1c measurements. The treatment history factors were the number of regimens the patient had tried; the number of visits since starting the current regimen; whether or not the patient had been previously prescribed metformin; and the patient's current regimen.

The prediction step of our algorithm is best illustrated through an example. Suppose we would like to estimate a patient's potential outcome under metformin monotherapy. To identify the importance of each factor in predicting outcomes, we used patient visits in which metformin monotherapy was prescribed to train an ordi-

Feature	NEWPT	NORX	MET0	ORAL0	INS0	MET1	METINS0	INS1	ORAL1	MET2	METINS1	INS2	MULTI
Male sex	0.01	0.01	<0.01	0.01	0.01	<0.01	0.01	0.01	0.03	0.01	<0.01	<0.01	0.01
Black race/ethnicity	0.01	0.01	<0.01	<0.01	0.01	0.01	0.01	0.02	0.01	<0.01	0.03	0.10	<0.01
Hispanic race/ethnicity	0.01	0.01	<0.01	0.01	0.01	0.01	0.01	0.02	<0.01	<0.01	0.02	0.07	<0.01
White race/ethnicity	<0.01	0.01	<0.01	0.02	0.02	<0.01	0.01	0.01	<0.01	0.01	0.04	0.08	0.02
Age	<0.01	<0.01	0.01	0.01	0.03	0.04	0.02	0.01	0.03	0.01	0.02	0.04	<0.01
Time since first diabetes treatment	0.07	0.03	0.03	0.04	0.01	0.02	0.03	0.03	0.03	0.01	0.02	0.02	<0.01
Most recent HbA1c	0.21	0.31	0.22	0.24	0.25	0.17	0.31	0.15	0.11	0.09	0.14	0.05	0.07
Frequency of HbA1c measurements	0.02	0.01	0.01	0.02	0.02	0.02	0.03	0.02	0.01	<0.01	0.02	0.01	0.01
Second most recent HbA1c	0.01	0.07	0.01	0.01	0.04	0.01	0.02	0.01	0.02	0.04	<0.01	0.01	0.01
Median HbA1c (last 1000 days)	0.05	0.11	0.02	0.07	0.01	0.08	<0.01	0.04	0.08	0.09	0.04	0.08	0.10
75th percentile HbA1c (last 1000 days)	<0.01	0.12	0.10	0.09	0.07	0.07	0.07	0.02	0.13	0.13	0.06	0.02	0.05
25th percentile HbA1c (last 1000 days)	0.06	0.09	0.04	0.02	0.03	0.03	0.08	<0.01	0.18	0.14	0.07	0.01	0.06
Mean HbA1c (last 1000 days)	0.18	0.04	0.06	0.04	0.07	0.07	0.01	0.09	0.08	0.22	0.03	0.04	0.15
Most recent BMI	0.03	0.07	0.05	0.04	0.01	0.05	<0.01	0.01	0.03	0.03	0.01	0.02	0.03
Median BMI (last 1000 days)	0.04	<0.01	0.07	0.02	0.04	0.03	0.04	0.03	<0.01	0.04	0.04	0.04	0.08
75th percentile BMI (last 1000 days)	0.07	0.03	0.04	0.01	0.05	0.05	0.02	0.08	<0.01	0.01	0.13	0.03	0.01
25th percentile BMI (last 1000 days)	0.09	0.01	0.03	0.04	0.03	0.05	0.03	0.14	0.07	0.02	0.05	0.06	0.09
Mean BMI (last 1000 days)	0.14	0.03	0.11	0.07	0.01	0.08	0.01	0.17	0.11	0.05	0.21	0.14	0.20
Ever prescribed metformin previously	<0.01	<0.01	<0.01	0.02	0.01	<0.01	0.01	0.02	<0.01	<0.01	0.01	0.02	0.01
Serum creatinine level >= 1.5 mg/dL	<0.01	0.01	0.01	0.01	0.02	<0.01	0.01	0.01	<0.01	<0.01	0.02	0.01	0.01
Number of lines of therapy tried	<0.01	0.01	0.01	0.02	0.03	0.03	0.06	0.01	0.01	0.02	0.01	0.06	0.02
Number of visits since starting current line of therapy	<0.01	<0.01	0.01	0.02	0.01	0.02	0.02	0.02	0.01	0.03	0.03	0.05	0.05
Patient currently prescribed metformin	<0.01	0.02	0.04	0.05	0.12	0.04	0.03	0.03	0.01	0.01	0.01	0.01	0.01
Patient currently prescribed insulin	<0.01	<0.01	0.11	0.01	0.05	0.06	0.05	0.01	0.05	0.03	0.01	0.04	<0.01
Number of unique drugs in current regimen	<0.01	0.01	0.01	0.10	0.04	0.07	0.11	0.05	<0.01	<0.01	0.01	0.01	0.01

Figure 4-3: Feature weights used to calculate similarity between patient visits.

Darker shading indicates larger values.

nary least squares regression on normalized values of each patient factor listed above. The most predictive factors were: the patient's most recent HbA1c measurement (regression coefficient magnitude 0.22), whether the patient was currently prescribed insulin (0.11), the patient's mean BMI over the past 1000 days (0.11), and several other HbA1c and BMI measurements (coefficient magnitudes ranging from 0.03 to 0.10); see Figure 4-3 for full details. To estimate the patient's potential outcome, we used the coefficient magnitudes to weight the Euclidean distance between this patient visit and each patient visit in which metformin monotherapy was prescribed. Thus, for any choice of k, we could rank the k closest neighbors from this treatment group. This procedure was repeated for each therapy in the patient's menu of treatment options.

Intuitively, the number of neighbors k used to estimate post-treatment HbA1c levels should increase with the size of the dataset. For each treatment t, we found the value k_t^* that minimized the root-mean-square error of the kNN predictions on a subset of the data not used to evaluate the algorithm. We regressed k_t^* on $\sqrt{n_t}$, and thus derived the dependence function $k_t^* = 0.34 \cdot \sqrt{n_t}$, which was used to select k in the prescriptive algorithm.

To verify the accuracy of the kNN HbA1c predictions, we evaluated the R^2 metric. Positive values of R^2 suggest patient characteristics are predictive of future HbA1c. For comparison, we evaluated the predictive accuracy of LASSO regression [90] and random forest [28], two state-of-the-art machine-learning methods used widely due to their high prediction accuracy. We used the predictions from these models in two alternative prescriptive algorithms.

4.2.5 Model evaluation

To evaluate the performance of the kNN-based prescriptive model, we tested the algorithm's recommendations on a set of patient data that had not been used when training the models.

Because counterfactual treatment effects are not observable, we used the weighted matching approach embedded in the kNN regression to impute potential outcomes,

an approach commonly used for causal inference in observational studies when randomization is unavailable [58]. For each visit, we applied our prescriptive algorithm to recommend a therapy. If that recommendation matched the prescribed standard of care therapy, we observed the true effect from the therapy. Otherwise, the outcome was imputed by averaging the outcomes of the most similar patient visits at which the recommended therapy was administered; these similar visits were chosen from a test set not used for training, and the number of neighbors k_t^* was selected to fit the size of the test set. This estimated outcome was compared to the true outcome under standard of care at the given patient visit.

Our hypothesis was that the average predicted HbA1c outcome after applying our prescriptive algorithm would be less than that observed from administering standard of care, resulting in a net average improvement in outcomes.

4.2.6 Sensitivity analysis

To ensure the evaluation of our algorithm was not sensitive to the particular random split of the database into training and test data, we evaluated the effectiveness of our algorithm (with fixed threshold $\delta = 0.8$) under additional random splittings of the data.

4.2.7 Software

All analyses were performed in R 3.3.0 [88].

4.3 Results

The R^2 of the kNN predictions on unseen data ranged from 0.20 to 0.54 depending on the regimen (Table 4.3). The strongest models were for insulin monotherapy, metformin monotherapy, metformin plus insulin, and multi-drug (3+) therapies. The R^2 values from the LASSO and random forest models ranged from 0.24 to 0.53. The predictive power was similar across the three methods.

Table 4.3: Out-of-sample R^2 under various predictive methods.

Regimen	**kNN**	**LASSO**	**Random Forest**
NEWPT	0.38	0.33	0.41
MET0	0.46	0.42	0.48
INS0	0.54	0.53	0.53
OTHER0	0.40	0.39	0.40
MET1	0.42	0.39	0.42
METINS0	0.46	0.46	0.47
INS1	0.44	0.43	0.43
OTHER1	0.34	0.35	0.35
MET2	0.32	0.32	0.33
METINS1	0.41	0.42	0.45
INS2	0.20	0.31	0.24
MULTI	0.46	0.36	0.46

The performance of the prescriptive algorithm is summarized in Table 4.4. The mean HbA1c outcome after treatment was 0.14% lower under the prescriptive algorithm than under the standard of care treatment, with standard error (SE) 0.01% and significance level $p \ll 0.001$. Of the 48,140 patient visits in our dataset, the algorithm differed from the standard of care for 15,323 visits, 31.8% of all visits. For this subset of visits, the mean HbA1c outcome under the algorithm was lower by $0.44 \pm 0.03\%$ compared with standard of care, with $p \ll 0.001$, a reduction from 8.37% under the standard of care to 7.93% under our algorithm. The median outcome for these visits was 0.21% lower under the prescriptive algorithm compared with standard of care. For comparison, the median difference for all visits was zero because, for 68.2% of visits, there was no difference between the algorithm's recommendation and the standard of care.

In our analysis, the mean difference in HbA1c was more negative than the median due to a left-skewed distribution. Some patients received particularly large benefits from using the prescriptive algorithm, which had an outsize effect on the mean but did not affect the median.

Figure 4-2 depicts the number of patients for whom the prescriptive algorithm recommended switching from a given current line of therapy to a given new line of therapy, along with the mean reduction in HbA1c for patient visits in each category.

Table 4.4: Performance of prescriptive algorithms.

All patient visits ($N = 48,140$)			
	kNN	**LASSO**	**Random Forest**
Mean HbA1c benefit relative to standard of care, % (SE)	-0.14 (0.01)*	-0.13 (0.01)*	-0.07 (0.01)*
Visits for which algorithm's recommendation differed from observed standard of care			
	kNN	**LASSO**	**Random Forest**
Number of visits (%)	15,323 (31.8%)	12,684 (26.3%)	14,302 (29.7%)
Mean HbA1c benefit relative to standard of care, % (SE)	-0.44 (0.03)*	-0.45 (0.03)*	-0.26 (0.03)*

* $p \ll 0.001$.

Among trajectories with at least 300 patients, the largest benefit of the algorithm was achieved through personalized recommendations for 7,564 patients currently on insulin monotherapy to switch to monotherapy with metformin or another blood glucose regulation agent. However, for the vast majority of patients currently on insulin-based regimens, the algorithm recommends that those patients continue with that therapy. Among the 7,564 patient visits, those who were recommended to switch from insulin were on average younger (mean age 52.9 years versus 61.4 years) and had substantially higher average HbA1c (11.0% versus 8.0%).

The performance of the prescriptive algorithm in specific patient subgroups is summarized in Tables 4.5 and 4.6. The overall mean HbA1c outcome using the prescriptive algorithm was 0.14% lower than standard of care for both male and female patients. The benefit of using the algorithm was 0.14% for black patients (29,120 visits), 0.09% for white patients (7,444 visits), 0.22% for Hispanic patients (6,732 visits), and 0.11% for all other patients (4,844 visits). The benefit of the algorithm was 0.20% for patients under the age of 60 and 0.08% for patients aged 60 or older. The benefit was 0.20% for patients with poor glycemic control, i.e. current

HbA1c greater than 7.0% as compared with 0.05% for those with good glycemic control.

Table 4.5: Performance of algorithm in study subgroups; all patient visits.

Subgroup	**Number of visits***	**Mean HbA1c benefit relative to standard of care, % (SE)†**
Male	20,231	-0.14 (0.01)
Female	27,909	-0.14 (0.01)
Black	29,120	-0.14 (0.01)
White	7,444	-0.09 (0.01)
Hispanic	6,732	-0.22 (0.01)
Other	4,844	-0.11 (0.01)
Age < 60	23,705	-0.20 (0.01)
Age 60+	24,435	-0.08 (0.00)
Good glycemic control (HbA1c ≤ 7)	18,156	-0.05 (0.01)
Poor glycemic control (HbA1c > 7)	29,984	-0.20 (0.01)

* $N = 48,140$
† $p \ll 0.001$ for all instances.

Our methodology motivates a provider dashboard that would report information on the demographics, medical history, and response to treatment for patients similar to an index patient. A prototype dashboard visualization for one sample patient visit is shown in Figure 4-4. The dashboard would include the patient's demographic and health information along with visualizations of the patient's treatment history and HbA1c progression. In addition, the dashboard would display the mean, standard deviation, and full distribution of HbA1c outcomes among the k_t^* nearest neighbors who received each treatment in the menu of options. Based on this evidence, the dashboard would display a treatment recommendation. The provider would have the ability to override this recommendation given any special management needs of the patient. For instance, if the patient is elderly and the distribution of HbA1c outcomes indicates that the recommended therapy has an elevated risk of hypoglycemia, the provider may opt for an alternative treatment.

Table 4.6: Performance of algorithm in study subgroups; visits for which algorithm's recommendation differed from standard of care.

Subgroup	**Number of visits**	**Percent of all visits in subgroup**	**Mean HbA1c benefit relative to standard of care, % (SE)***
Male	6,363	31.5%	-0.44 (0.02)
Female	8,960	32.1%	-0.44 (0.02)
Black	9,103	31.3%	-0.45 (0.02)
White	2,309	31.0%	-0.29 (0.03)
Hispanic	2,400	35.7%	-0.61 (0.03)
Other	4,844	31.2%	-0.34 (0.04)
Age < 60	8,783	37.1%	-0.55 (0.02)
Age 60+	6,540	26.8%	-0.30 (0.02)
Good glycemic control (HbA1c ≤ 7)	4,438	24.4%	-0.20 (0.02)
Poor glycemic control (HbA1c > 7)	10,885	36.3%	-0.54 (0.02)

* $p \ll 0.001$ for all instances.

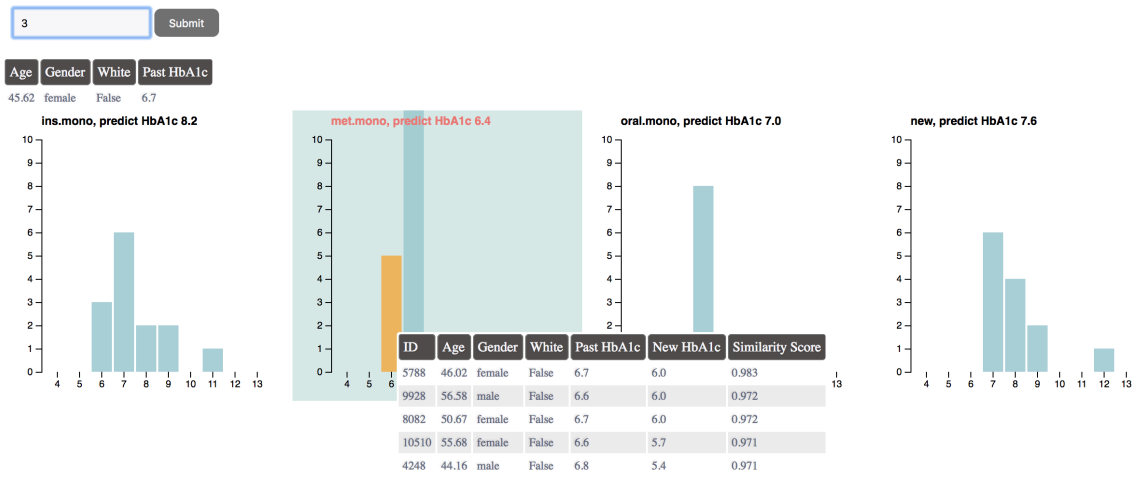


Figure 4-4: Visualization of prescriptive algorithm: Provider dashboard prototype.

This figure visualizes how the prescriptive algorithm can be used by providers for a single patient.

- The provider enters the patient id of the current patient in the top left textbox.
- The table below shows basic summary of patient demographics and medical history. We envision the time series of patient HbA1c levels and past medications will also be presented visually here.
- Each plot represents a scenario under a potential treatment, providing the predicted post-treatment HbA1c. The algorithm's recommended treatment is highlighted.
- For a given treatment, the histogram represents the distribution of actual HbA1c outcomes under the k_t^* most similar patient visits in the data set, the post-treatment HbA1c level is on the horizontal axis and the number of visits is on the vertical axis.
- When the provider mouse-over the bars in the histogram, more details on the similar patient visits are displayed together with a similarity score for the provider to further inspect.

The overall mean HbA1c outcome using the LASSO-based prescriptive algorithm was lower by $0.13 \pm 0.01\%$ ($p \ll 0.001$) compared with the mean standard of care outcome. The benefit from using the random-forest-based prescriptive algorithm relative to standard of care was $0.07 \pm 0.01\%$ ($p \ll 0.001$).

In the sensitivity analyses, under three alternate random splittings of the dataset, the overall mean benefit of using the prescriptive algorithm compared with standard of care ranged from 0.11% to 0.15% ($p \ll 0.001$ in all instances).

4.4 Discussion

To our knowledge, we present the first prescriptive method for personalized type 2 diabetes care. Using historical data from a large EMR database, this novel prescriptive method resulted in an average HbA1c benefit of 0.44% at each doctor's visit for which the algorithm's recommendation differed from standard of care.

Our method incorporates patient-specific demographic and medical history data to determine the best course of treatment. Compared to other machine learning methods considered, the kNN prescriptive approach is highly interpretable and flexible in clinical applications. The novelty of our approach is in personalizing the decision-making process by incorporating patient-specific factors. This method can easily accommodate alternative disease management approaches within specific subpopulations, such as patients with chronic kidney disease and elderly patients. We believe this personalization is the primary driver of benefit relative to standard of care.

In practice, the algorithm can be integrated into existing EMR systems to dynamically suggest personalized treatment paths for each patient based on historical records. The algorithm ingests and analyzes EMR data and generates recommendations. An intuitive, interactive dashboard summarizes the evidence for the recommendation, including the expected distribution of outcomes under alternative treatments (Figure 4-4).

Due to the nature of retrospective data from existing EMR, this study has several limitations. Patients were not randomized into treatment groups. While our matching

methodology controls for several confounding factors that could explain differences in treatment effects, we can only estimate counterfactual outcomes. EMR data do not include socio-economic factors or patient preferences that may be important in treatment decisions. Due to lack of sufficient data, GLP-1 agonists were not considered as a separate drug class. If more data were available, we could further differentiate regimen types beyond the thirteen we include in this analysis. In addition, the study population from BMC may not be representative of the US population as a whole.

With EMR medication order data alone, we cannot be certain whether a prescribed medication was filled or taken, and cannot know precisely when the medication was stopped. Although this data quality issue could hamper attempts to make drug efficacy comparisons, our analysis aims to address the question of which drugs to prescribe under real-worlds scenarios. We optimize for an outcome that takes into account unobserved factors such as non-adherence. For instance, if non-adherence is more prevalent among patients prescribed insulin than other regimens, this perspective may explain why, in our study population, the algorithm recommends insulin less often than it is prescribed in clinical practice.

Our method can be extended to be more flexible and comprehensive. Currently the prescriptive algorithm does not support individualized glycemic targets; we assume that a lower glycemic level is always preferred. The study currently optimizes only for a single health outcome; a more comprehensive algorithm would consider adverse event outcomes as well.

Despite these limitations, the study establishes strong evidence of the benefit of individualizing diabetes care. The success of this data-driven approach invites further testing using datasets from other hospital and care settings. Testing the prescriptive algorithm in a clinical trial setting would provide even stronger evidence of clinical effectiveness. As large-scale genomic data becomes more widely available, the algorithm could readily incorporate such data to reach the full potential of personalized medicine in type 2 diabetes.

In this study, we developed a novel data-driven prescriptive algorithm for type 2 diabetes that improves significantly on the standard of care when tested on patient-

level EMR data from a large medical center. Our work is a key step toward a fully patient-centered approach to diabetes management.

Chapter 5

An Actionable Tool for Mortality Predictions in Cancer Patients

This work is in revision with Journal of Clincal Oncology: Clinical Cancer Informatics, with co-authors Dimitris Bertsimas, Jack Dunn, Colin Pawlowski, John Siberholz, Alexander Weinstein, Eddy Chen and Aymen Elfiky [8].

With rapidly evolving treatment options in cancer, the complexity in the clinical decision-making process for oncologists represents a growing challenge magnified by oncologists' disposition of intuition-based assessment of treatment risks and overall mortality.

Given the unmet need for accurate prognostication with meaningful clinical rationale, in this chapter we developed a highly interpretable prediction tool to identify patients with high mortality risk prior to the start of treatment regimens. Using electronic health record (EHR) data between 2004 and 2014 from a large national cancer center, we built an actionable tool using novel development in modern machine learning to predict 60-, 90- and 180-day mortality from the start of an anticancer regimen. Our proposed prediction models achieved significantly higher estimation quality in unseen data (AUCs: 0.83-0.86) compared to benchmark models. We identified key predictors of mortality such as change in weight and albumin levels. The results are presented in an interactive and interpretable tool

(`stuff.mit.edu/~zhuo/tree_vis/index.html`).

Our fully-transparent prediction model was able to distinguish with high precision between highest and lowest risk patients. Given the rich data available in EHRs and advances in machine learning methods, this tool can have significant implications for value-based shared decision making at the point-of-care and personalized goals-of-care management to catalyze practice reforms.

5.1 Introductions

As the landscape of treatment options continues to expand for cancer patients with the development of novel targeted therapies and immunotherapies, the increasing complexity in the treatment decision making process for physicians alike has represented a growing challenge. When considering a new line of treatment for a patient, a clinician aims to account for a number of factors including performance status, previous treatments, the toxicity and efficacy of treatment at hand, and the patient's overall goals of care. Most significantly, in the setting of non-curative-intent treatments, the clinician needs to make assessment on the benefits of therapy versus the risk of mortality given patient characteristics. Today such assessments are made largely using human intuition and experience, often unable to take into full account the past and present trends and nuances of a patient's objective clinical and disease trajectories over time. Factually, physicians tend to overestimate prognosis in cancer [51, 86], and patient preferences are very sensitive to these estimates [30, 84, 55]. Agencies including the National Quality Forum (NQF) and organizations such as the American Society of Clinical Oncology (ASCO) have identified chemotherapy administration to patients for whom there is no clinical benefit as the most pervasive and superfluous practice in oncology [46, 82]. Indeed, unqualified use of aggressive treatments in progressive disease is associated with increased symptom burden for patients, aggregate adverse events, and intensity and cost-of-care interventions that have little morbidity or mortality benefit [97, 39, 44, 52].

In this chapter, using the electronic health records (EHRs) of the spectrum of

all cancer patients treated at a tertiary cancer center and novel machine learning algorithms constructed by the coauthors, we developed a predictive tool (delivered as a web or mobile-based application) to estimate the probability of mortality for a particular patient and a particular envisioned cancer treatment. We refer the reader to the website (`stuff.mit.edu/~zhuo/tree_vis/index.html`) for the tool. The tool we developed makes an important contribution to the clinical practice of oncology as it is:

1. **Personalized and specific:**

   The tool takes as inputs the EHR of a particular patient, the cancer type and an envisioned cancer treatment and outputs the mortality risk adjusted for these patient characteristics.

2. **Interpretable and clinically meaningful:**

   A physician can easily understand the reasoning behind the algorithm, illustrated as an interpretable decision tree. The model also identifies key predictors of mortality such as change in weight.

3. **Evidence-based and data-driven:**

   The tool was informed by EHRs of more than 23,000 patients at a large national cancer hospital. We included 401 predictors including demographics, medical and treatment history, laboratory tests, and genomic results.

4. **Actionable:**

   The clinician can compare different envisioned treatments for a particular patient with respect to the range of mortality risk and make decisions informed by these estimates.

5. **Validated and accurate:**

   We compare the accuracy and the area under the curve (AUC) in unseen patient data from 2012-2014, with very encouraging results compared to competing approaches.

6. **Based on novel development in modern machine learning:**
   The methodology of this chapter is based on two novel algorithms: a) the predictive tree developed using optimization ideas, [7] and b) the statistical method for missing data imputation as in Chapter 2 [18].

Because of its importance in clinical decision making, mortality prediction has long attracted research interest in the medical and biostatistical communities. In recent years, some prediction models have been made accessible online as tools, which has allowed the results from prognostic research to be readily available for use at point of care. Although such tools provide convenience to physician users, the data and models behind these tools limit the potential for reaching higher prediction quality while maintaining the interpretability. Most tools are based on cancer registry patient cohorts, limited to a small number of patient characteristics from the time of diagnosis, thereby overlooking critical information over a period of time that ultimately has significant implications for a patient's prognosis. Even with more detailed datasets, the interface of the tool requires learning accurate models with few variable inputs; black-box models such as artificial neural networks or gradient boosted trees that rely on a large number of predictors would not be suitable for such interactive tools. Further, those methods offer little explanation to the physicians on why such predictions were made. These concerns highlight critical considerations that explain the lack of practical translation of novel prognostic research into standard clinical workflows.

In notable contrast to the previous efforts, in this work we built a highly interpretable tool to predict individual mortality risk for a given treatment, using rich EHR data of more than 23,000 patients from a major academic cancer center with over 400 predictors. The remainder of the chapter details the data, method, and validation results of the model.

5.2 Methods

We retrospectively obtained patient data from the EHR and linked Social Security Administration mortality data for patients at the Dana-Farber/Brigham and Women's Cancer Center (DF/BWCC) from 2004 through 2014. To be eligible for the study, patients must be at least 18 years of age at cancer diagnosis, and have had at least one anti-cancer treatment over the course of their care. The primary outcomes were mortality rates at 60, 90 and 180 days after initiation of anti-cancer regimen, including chemotherapy, immunotherapy, and targeted therapy. If the patient's date of death is missing and the last known date alive is before the cutoff, that patient's record is censored for predicting that outcome. Each observation corresponds to a patient initiating an anti-cancer regimen, which was systematically recorded in the EHR.

5.2.1 Data

We considered 401 predictive features, including demographics, cancer diagnoses, comorbidities, prior treatments, resource utilization, gene mutations, and vital signs and laboratory tests results. Missing values were imputed using the algorithm Optimal-Impute developed by selected coauthors,13 which frames the imputation task as a family of optimization problems and solves directly. This algorithm has demonstrated significant improvement in downstream prediction tasks compared to classical methods. We used regimens initiated from 2004-11 as the training set, and then assessed each model's predictive performance using regimens initiated from 2012-14 as the validation set. The institutional review boards of Dana Farber Cancer Institute, Partners Healthcare, and Massachusetts Institute of Technology approved this study.

5.2.2 Model and Tool Development

The mortality predictions are based on a novel decision tree algorithm, Optimal Classification Trees.12 We selected decision trees for their advantage in interpretability,

where the predictions are based on a few decision splits on variables of high importance. Such tree structure can readily model nonlinearities and interaction between variables. However, classical decision tree methods typically cannot achieve the same level of accuracy as their less interpretable counterparts such as artificial neural networks and gradient boosted trees. To mitigate the tradeoff between interpretability and prediction accuracy, we made use of recent advances in machine learning. In particular, the Optimal Classification Trees algorithm trains a single decision tree with state-of-the-art performance, achieving high accuracy without the need to sacrifice interpretability. After training the model, for interpretation and verification, we generate the following: 1) a visualization of the decision tree, where experts can follow through the logic and predicted mortality risks to verify the clinical relevance; and 2) feature importance scores, which provide an estimate on the relative importance of the key variables in mortality predictions.

We next built the interactive tool for physicians based on the trained decision trees. The tool is made available as a web-based application (`stuff.mit.edu/~zhuo/tree_vis/index.html`), in the format of a patient characteristics questionnaire. The clinician user will have the option to select 60-, 90-, or 180-day mortality predictions and cancer types (all cancers, breast, lung, ovarian, kidney, etc.). This user will then be prompted to provide answers to a few questions, corresponding to the decision splits in the trained decision tree. The questions are adaptive based on the user's answers to previous questions. Once answers are provided, the tool will generate a predicted mortality risk specific to this patient. Examples are provided in the results section.

We also envision this tool to be integrated in an EHR environment, where the answers to the questions are automatically pulled from the EHR database, and the physician will simply have to verify the inputs and view the output mortality risks in the workflow.

5.2.3 Performance Comparisons

We evaluate the performance of Optimal Classification Trees in unseen patient data for the quality of mortality predictions. To demonstrate our improvement in prognostic quality, our model should be compared against the established prognostic studies on the same patient population. For such comparisons, the outcome variables (short-term mortality) need to be aligned, and the predictor variables from prior studies need to be available. Unfortunately, among the studies we identified, no such comparison could be conducted. As proxies, a variety of other machine learning models were trained and validated on the same set of data for comparisons.

First, a multivariate logistic regression model was trained to predict mortality on a subset of variables commonly used in prior studies. Specifically, these variables included demographic information, cancer characteristics, and comorbid. Laboratory test result history and gene mutations were excluded as they were not widely used in prior studies. We regard this model as the baseline model given it reflected the data and methodology used in previous, classical prognostic algorithms. Then, we added in the full set of potential predictors from the EHR and fit both a logistic regression model and an elastic net regularized logistic regression to predict mortality. We also trained a classical classification and regression trees (CART) algorithm as a baseline for decision tree based methods [29]. Finally, as a comparison against modern black box algorithms, we fitted gradient boosted trees using the full set of potential predictors extracted from the EHR [43]. All model parameters were selected via cross-validation, a procedure where model performance on a reserved validation set of patients are used to select best parameters.

We report AUC for each model and prediction accuracy based on a default 50% threshold. The analyses were repeated for each of the three time horizons (60-, 90-, and 180-day mortality) and subgroups by cancer type.

5.3 Results

5.3.1 Patient and Treatment Characteristics

A total of 23,983 patients were selected in the cohort. Among those, 14,427 (60.2%) were in the training set and 9,556 (39.8%) were in the validation set. These patients initiated 46,646 total new regimens; 2,619 (5.6%) of regimen starts were followed by the patient's death within 60 days and 44,027 (94.4%) regimens were not. Overall, regimens that led to mortality were associated with more severe diseases and heavier resource utilization before initiating the regimen. For example, higher proportions of lung cancer and pancreatic cancer patients were observed in regimens that led to mortality. Patients in this group on average also had more severe disease staging, as well as higher number of prescribed medications, inpatient/outpatient visits, and blood infusions. The regimens that led to mortality were also associated with more comorbid conditions such as congestive heart failure, stroke, diabetes. Finally, the laboratory test results for regimens that led to mortality were often significantly worse than the ones that did not lead to mortality (lower weight and albumin, higher cancer marker, worse blood counts, etc.). The median survival was 514 days for all patients.

5.3.2 Interpretable Tool Based on Machine Learning

We trained the Optimal Classification Trees to predict the 60-, 90-, and 180-day mortality. The model produced accurate predictions with accuracies of 94.9%, 93.3%, and 86.1%, and AUCs of 0.86, 0.84, and 0.83, respectively. We further trained the model to predict the mortality for each subgroup of cancer sites, achieving similarly high estimation qualities with AUCs ranging from 0.77 to 0.90. Based on the prediction algorithm, we developed the tool and made it available online at `stuff.mit.edu/~zhuo/tree_vis/index.html`, a screenshot of which is illustrated in Figure 5-1. Once the clinician enters the desired time horizon and cancer site, the clinician will answer a few questions regarding the patient before the tool outputs a

final risk estimation. In this example, in learning the risk of 60-day mortality for a lung cancer patient, the tool adaptively asks the clinician to input the percent change in weight and the albumin levels. With these two input values, the tool immediately presents the final risk for 60-day mortality of 46.79%.

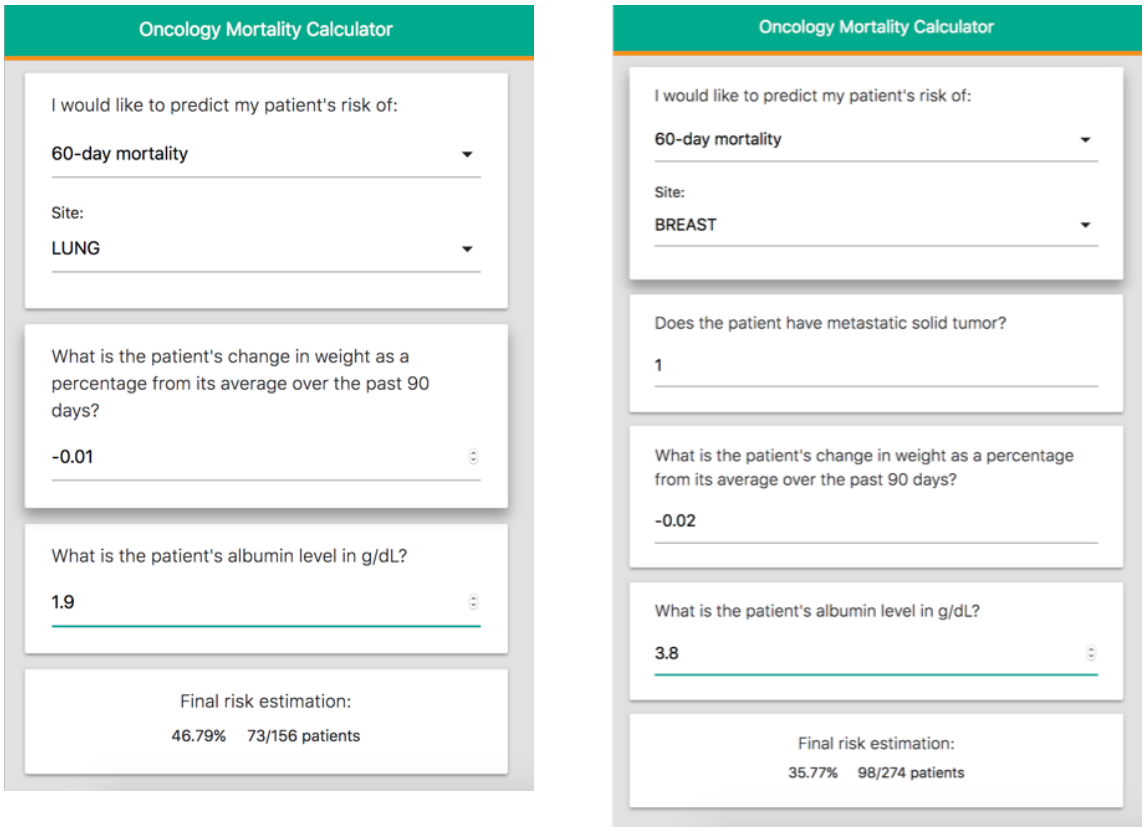


Figure 5-1: Screenshots of the cancer mortality prediction questionnaire. The clinician enters responses to a few questions regarding the patient's medical history, many of which can be automatically populated via EHR integration, and the tool will immediately generate predictions.

5.3.3 Model Interpretation

Each model trained with Optimal Classification Trees presents a highly interpretable decision tree which the tool is based on. Such tree stratifies patients into risk groups based on values of a sequence of key variables, the selection of which is learned

automatically by the model. As an example, Figure5-2 presents the tree that predicts the 60-day mortality for breast cancer patients. The total number of metastatic solid tumors is the first splitting variable - patients with few metastatic solid tumors (left of the tree) are placed into the lower-risk branch (0.4%) compared to the higher-risk one (5.84%). Within each branch, patients are further stratified based on other variables. For example, in the lower-risk branch, if the pulse is above 86, the patient has a mortality risk of 1.31%; if the pulse is below 86, depending on the age (below or above 60 years), the risk is 0% and 0.88%, respectively. In the higher-risk branch, many of the decisions are based on change in weight, albumin, and some other laboratory test results.

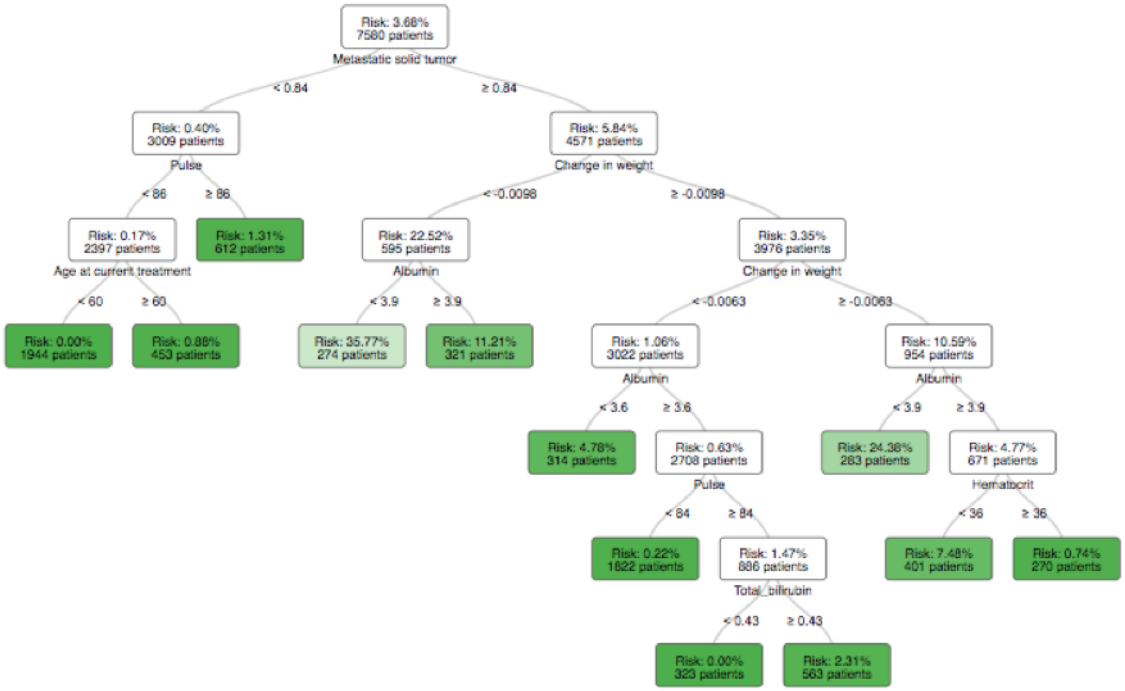


Figure 5-2: Mortality prediction tree for breast cancer patients on 60-day mortality. Patients are stratified based on a sequence of variables, eventually placed into a mortality risk bin.

This model also produces feature importance rankings in mortality prediction (Figure 5-3). In tree based models, the feature importance score measures the relative

contribution of a particular feature in the model based on the frequency this variable is selected for splitting and the improvement in model performance at each split; the score of all variables sum up to one. Among all patients, percent change in weight from a patient's moving average over the past 90 days is the most important predictive feature of mortality (more drastic decrease is associated with higher risk of mortality). Albumin level, pulse, white blood counts, total bilirubin, and weight were the next predictive variables.

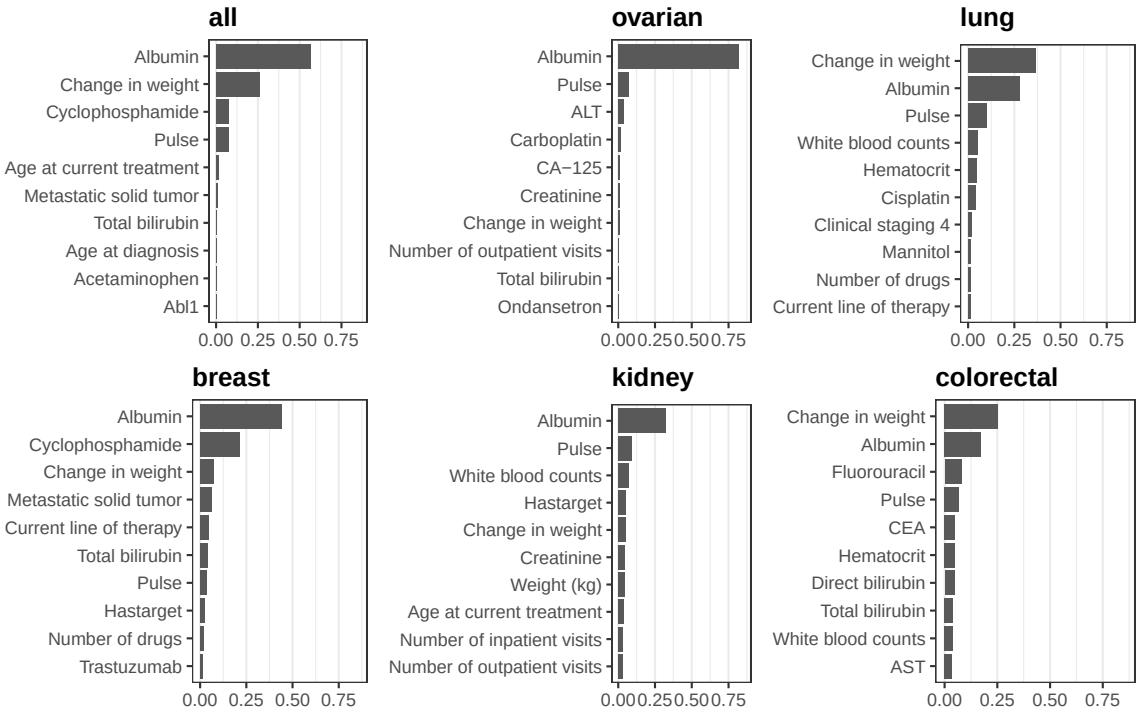


Figure 5-3: Feature importance in 60-day mortality prediction for patients (all cancer and by cancer sites). The importance score is based on the relative contribution to the model performance of each feature during the Optimal Classification Trees training process. The ten most important predictors are shown in this figure.

While the feature importance does not demonstrate the interaction across predictive variables, the relationship was evidently characterized by the tree structure. Because the change in weight was suggested in many cancer types as one of the top predictors of mortality, we investigate further the impact of change in weight as a sin-

gle variable on the mortality outcomes. Figure 5-4 presents the 60-day, 90-day, and 180-day observed mortality rates stratified by the weight change. In general, more weight loss is associated with higher mortality rates, with the exception of the highest quantile (weight gain of at least 1.15%). We suspect that there are confounding factors affecting the weight gain for patients at moderate mortality risk as a result of edema. In fact, lower albumin levels are observed in the highest weight gain group (see figure legend). This evidence further suggests that non-linear interactions modeled by decision trees are suitable for these complicated relationships. Aside from this group, the trend is generally monotone, with the risk of mortality highest in the group with at least 2.62% weight loss, almost doubling that in the next risk group.

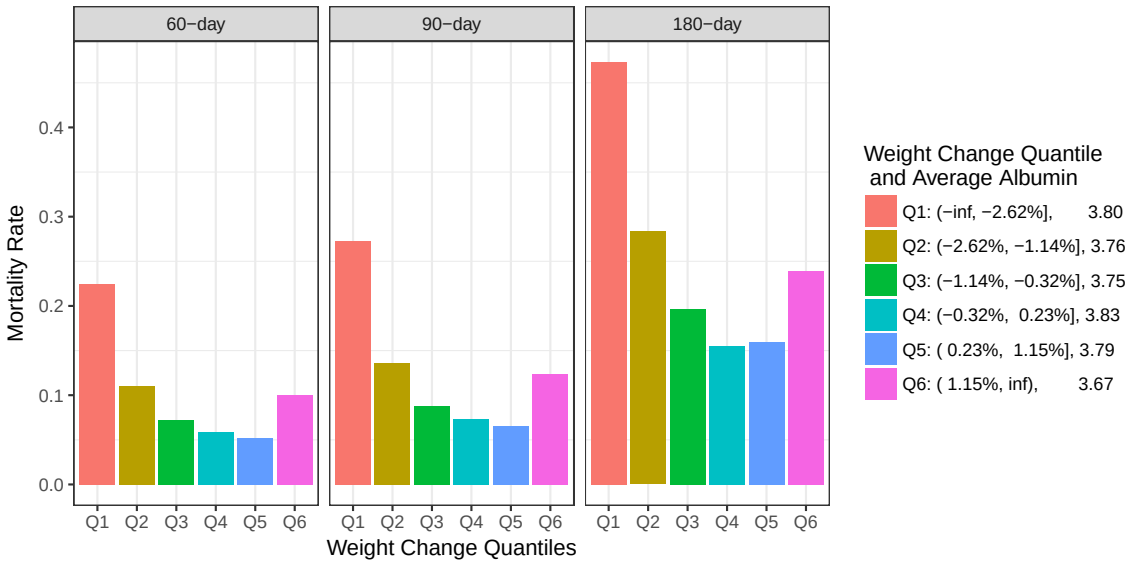


Figure 5-4: Mortality rates (60-day, 90-day, and 180-day) stratified by change in weight, as a percent of the past 90-day average. More weight loss is generally associated with higher mortality rates.

5.3.4 Machine Learning Models Comparison of Performance

Among all the transparent machine learning models being compared against, Optimal Classification Trees achieved the best performance in validation (Table 5.1). The only

Table 5.1: Model performances (accuracy and AUC) for 60-day, 90-day, and 180-day mortality predictions in the validation set, comparing Optimal Classification Trees against five other prediction models.

Mortality predictions	60-day	90-day	180-day
Accuracy			
Logistic regression (fewer predictors)	94.6%	93.0%	83.4%
Logistic regression	94.4%	93.0%	84.5%
Regularized logistic regression	94.9%	93.1%	84.6%
CART decision tree	93.8%	91.1%	83.7%
Optimal Classification Tree	94.9%	93.3%	86.1%
Gradient boosted trees	95.0%	93.6%	87.2%
AUC			
Logistic regression (fewer predictors)	0.74	0.76	0.76
Logistic regression	0.74	0.76	0.75
Regularized logistic regression	0.79	0.80	0.80
CART decision tree	0.81	0.79	0.78
Optimal Classification Tree	0.86	0.84	0.83
Gradient boosted trees	0.90	0.89	0.87

method that improves over Optimal Classification Trees is the black-box method of gradient boosted trees. The receiver operating characteristic (ROC) curves for each of the methods are in Figure 5-5. In terms of AUC, classical logistic regression, both with a smaller set of predictors or full set of predictors, has the lowest AUC compared to all other methods. CART and regularized logistic regression have improved out-of-sample performance than logistic regression yet weaker performance than Optimal Classification Trees and gradient boosted trees ($p < 0.001$). In the subgroup analyses, we found similar performance for our method. In the sensitivity analyses where different missing data imputation methods were used, the results did not change drastically (results not shown).

5.4 Discussions

Within the context of the growing momentum toward a value-based healthcare delivery system,6 we built an accurate prognostic tool to predict a cancer patient's survival probability before initiating a new treatment. Developed with the intent to augment physicians' clinical decision-making at the point-of-care, it also holds the

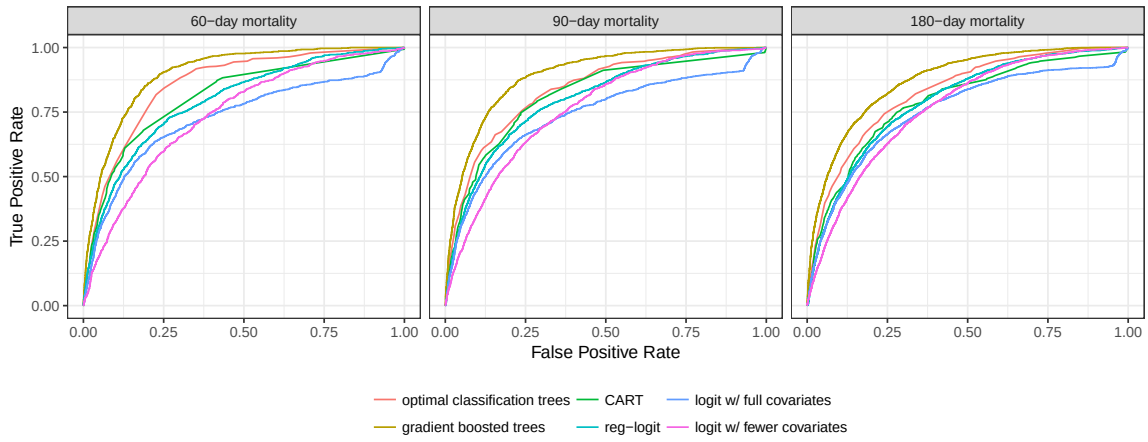


Figure 5-5: ROC curves for 60-day, 90-day, and 180-day mortality predictions, comparing the following methods: Optimal Classification Trees, CART, logistic regression with fewer variables, logistic regression with all variables, regularized logistic regression, and gradient boosted trees.

possibility of acting as a catalyst for necessary patient-provider discussions, leading to true informed consent and goals-of-care concordance.

Our predictive tool uses novel machine learning methods to support the movement toward fully personalized, evidence-based treatment decisions. The Optimal Classification Trees model we used is fully-transparent, interpretable, and produces highly accurate results. With simple question-and-answer interactions and potential EHR integration, the tool can become an essential component in an oncologist's workflow, augment clinical decision making, and prompt defined crucial conversations between the primary oncologist and the patient. In the back end, the novel Optimal-Impute method completes the necessary data cleaning process that eventually leads to accurate results in the downstream tasks of mortality prediction.

The rich data constitutes the other key reason for our high prediction quality. Because this study predicts mortality in a large population with available EHR data, we were able to study a much broader set of covariates than typical mortality prediction studies. We studied 401 covariates in total, including 289 variables encoding gene mutation results, 52 on recent treatments, and 18 on recent laboratory and vital

test results. The longitudinal nature of the EHR data used in this study further enables the study of patient characteristic through time. For instance, we included the percentage change in weight from the 90-day moving average weight measurement, a predictor not used in previous mortality prediction work.

We would like to perform head-to-head comparisons against existing prognostic studies and online tools. However, among the short-term mortality studies we have identified, most require subjective variables such as performance status and clinician assessment which our data do not contain. As proxies, we compared against surrogate models trained using the same set of patient data as our Optimal Classification Trees-based model. The stronger results under our model suggest that our data and method have an edge compared to traditional approaches.

Our study has several limitations. First, as a single-institution retrospective study, it is subject to data selection and measurement biases. Second, validation was performed using an internal data set. In addition to the internal validation we performed, the algorithm must be tested prospectively in an external cohort. Looking forward, it is necessary to anticipate that this model may become outdated as a result of novel drugs and changing treatment paradigms. As precision-based disease variables such as whole-genome sequencing data become available in more patients, our model performance could potentially improve. The process of iteratively fine-tuning and updating the prognostic model is essential for relevant integration into contemporary clinical practice. Further prospective studies with this and other types of tools should be done to demonstrate the clinical utility of this tool, measuring the impact on mortality and other outcomes such as decreased length of stay in the hospital, earlier enrollment to hospice, and self- or family-reported quality of life.

5.5 Conclusions

We have developed an actionable tool for individual mortality prediction among cancer patients prior to treatment initiation, based on an accurate prediction model by combining: 1) a large cohort of general cancer patients, 2) longitudinal EHR data that

provide extensive and nuanced information compared to registry and claims data, 3) novel machine learning methods with high interpretability and state-of-the-art performance, and 4) clinical intuition and knowledge from experienced oncologists on the team. Notably, this machine learning-based algorithm could be easily generalized across any institution with an EHR to catalyze, engage, and guide necessary physician-patient discussions about the care trajectory, including risks and rewards of treatment choices prior to treatment initiation. Moreover, the implications of this work demonstrate the capacity to actionably integrate EHR data from large cohorts of patients, advanced machine learning algorithms, and human medical expertise, which translate to more advances that improve clinical outcome and workflow efficiency in other areas of medicine and healthcare delivery.

Chapter 6

Conclusions

In this thesis, I took a new look at some of the classical problems in machine learning and statistics with a fresh perspective from optimization. The problems such as missing data imputation (Chapter 2) and classification (Chapter 3), have shown to benefit from the added edge of rigorous optimization formulations and solution techniques. Because of the flexibility introduced in the optimization framework, we were able to incorporate novel development in predictive methods such as *Optimal Trees* into the formulation of some decade-old problems.

These newly developed machine learning algorithms have not only demonstrated improved performance in large-scale real world data set experiments, but are also solving the problems in health care that motivated the development of these algorithms. In the case of mortality prediction in cancer patients (Chapter 5), applying `opt.impute` for missing data along with the cutting edge algorithm *Optimal Classification Tree* on a rich EHR data set with demographic, treatment and medical history, laboratory tests results, and genomic information, we were able to accurately risk stratify the patients and provide valuable insight to care providers, especially regarding end-of-life decisions to improve patient quality of life.

The story of how we approached personalized diabetes management in Chapter 4 showcases the power of prescriptive analytics that combines predictive machine learning and optimization in the health care setting. In our simulated clinical trial, by adapting treatment recommendations to different patients, we demonstrated a

clinically relevant reduction in average hemoglobin A1c levels compared to current practice. More importantly, to our knowledge this is the first study that takes a data-driven, personalized approach to recommend treatments for diabetic patients with statistical validations. We hope that it will encourage more researchers in operations research and machine learning to continue developing personalized medicine in other medical domains.

As the end goal is to influence clinical practice, I firmly believe that academic research alone is not sufficient. Physicians need readily accessible and understandable models that are integrated to their workflows. As prototypes, we present the mortality prediction as an online interactive tool, where the physician can quickly obtain the risk for a given patient by filling out a simple, adaptive questionnaire. We envision with EHR integration the process will be further streamlined. For personalized diabetes management, we present the prototype dashboard where the distribution of outcomes among similar patients under alternative treatments are plotted, providing the justification on why such particular treatment is recommended. In both cases, the physician can see clearly the rationale behind how the prediction or prescription is made. Combined with human judgment and intuition, we are hopeful that such clinical decision tools developed under our models will be able to take clinical practice to the next level, further improving the quality of care delivered.

Bibliography

[1] Aharon Ben-Tal, Sahely Bhadra, Chiranjib Bhattacharyya, and Arkadi Nemirovski. Efficient methods for robust classification under uncertainty in kernel matrices. *Journal of Machine Learning Research*, 13(10):2923–2954, 2012.

[2] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[3] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000.

[4] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[5] Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[6] Dimitris Bertsimas and Martin S. Copenhaver. Characterization of the equivalence of robustification and regularization in linear and matrix regression. *European Journal of Operational Research*, 2017.

[7] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, pages 1–44, 2017.

[8] Dimitris Bertsimas, Jack Dunn, Colin Pawlowski, John Siberholz, Alexander Weinstein, Ying Daisy Zhuo, Eddy Chen, and Aymen Elfiky. An actionable tool for mortality predictions in cancer patients. *Submitted for publication*, 2018.

[9] Dimitris Bertsimas, Jack Dunn, Colin Pawlowski, and Daisy Zhuo. Robust classification. *Submitted for publication*, 2017.

[10] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *arXiv preprint arXiv:1402.5481*, 2014.

[11] Dimitris Bertsimas, Nathan Kallus, Alexander Weinstein, and Ying Daisy Zhuo. Personalized diabetes management using electronic medical records. *Diabetes Care*, 40:210–217, 2017.

[12] Dimitris Bertsimas and Angela King. An algorithmic approach to linear regression. *Operations Research*, 64(1):2–16, 2015.

[13] Dimitris Bertsimas and Angela King. Logistic regression: From art to science. *Statistical Science*, 32(3):367–384, 2017.

[14] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016.

[15] Dimitris Bertsimas and Rahul Mazumder. Least quantile regression via modern optimization. *Annals of Statistics*, 42(6):2494–2525, 2014.

[16] Dimitris Bertsimas, Allison O'Hair, Stephen Relyea, and John Silberholz. An analytics approach to designing combination chemotherapy regimens for cancer. *Management Science*, 62(5):1511–1531, 2016.

[17] Dimitris Bertsimas, Allison K. O'Hair, and William R. Pulleyblank. *The Analytics Edge*. Dynamic Ideas LLC, 2016.

[18] Dimitris Bertsimas, Colin Pawlowski, and Ying Daisy Zhuo. From predictive methods to missing data imputation: An optimization approach. *Journal of Machine Learning Research*, 2018.

[19] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[20] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*, volume 6. Athena Scientific, Belmont, MA, 1997.

[21] Dimitris Bertsimas and Bart Van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *arXiv preprint arXiv:1709.10029*, 2017.

[22] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98.

[23] Chiranjib Bhattacharyya, KS Pannagadatta, and Alexander J. Smola. A second order cone programming formulation for classifying missing data. In *Neural Information Processing Systems (NIPS)*, pages 153–160, 2005.

[24] Jinbo Bi and Tong Zhang. Support vector classification with input data uncertainty. *Advances in neural information processing systems*, 17:161–169, 2005.

[25] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *ACML*, pages 97–112, 2011.

[26] Trond Hellem Bø, Bjarte Dysvik, and Inge Jonassen. LSimpute: accurate estimation of missing values in microarray data with least squares methods. *Nucleic Acids Research*, 32(3):e34–e34, 2004.

[27] Lígia P. Brás and José C. Menezes. Improving cluster-based missing value estimation of DNA microarray data. *Biomolecular Engineering*, 24(2):273–282, 2007.

[28] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[29] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and regression trees*. CRC press, 1984.

[30] M. D. Brundage, J. R. Davidson, and W. J. Mackillop. Trading treatment toxicity for survival in locally advanced non-small cell lung cancer. *Journal of Clinical Oncology*, 15(1):330–340.

[31] Lane F. Burgette and Jerome P. Reiter. Multiple imputation for missing data via sequential regression trees. *American Journal of Epidemiology*, 172:1070–1076, 2010.

[32] Stef Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3), 2011.

[33] Zhipeng Cai, Maysam Heydari, and Guohui Lin. Iterated local least squares microarray missing value imputation. *Journal of Bioinformatics and Computational Biology*, 4(05):935–957, 2006.

[34] Rich Caruana. A non-parametric EM-style algorithm for imputing missing values. In *AISTATS*, 2001.

[35] Centers for Disease Control and Prevention. Basics about diabetes. `http://www.cdc.gov/diabetes/basics/diabetes.html`, 2015. Accessed: 2015-04-16.

[36] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[37] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[38] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(Dec):265–292, 2001.

[39] A. de Gramont, A. Figer, M. Seymour, M. Homerin, A. Hmissi, J. Cassidy, C. Boni, H. Cortes-Funes, A. Cervantes, G. Freyer, D. Papamichael, N. Le Bail, C. Louvet, D. Hendler, F. de Braud, C. Wilson, F. Morvan, and A. Bonetti. Leucovorin and fluorouracil with or without oxaliplatin as first-line treatment in advanced colorectal cancer. *Journal of Clinical Oncology*, 18(16):2938–2947.

[40] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38, 1977.

[41] B. J. Druker, M. Talpaz, D. J. Resta, B. Peng, E. Buchdunger, J. M. Ford, N. B. Lydon, H. Kantarjian, R. Capdeville, S. Ohno-Jones, et al. Efficacy and safety of a specific inhibitor of the bcr-abl tyrosine kinase in chronic myeloid leukemia. *New England Journal of Medicine*, 344(14):1031–1037, 2001.

[42] Laurent El Ghaoui, Gert René Georges Lanckriet, and Georges Natsoulis. Robust classification with interval data. Technical Report UCB/CSD-03-1279, 10 2003.

[43] Jane Elith, John R. Leathwick, and Trevor Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77(4):802–813, 2008.

[44] Ezekiel J. Emanuel, Yinong Young-Xu, Norman G. Levinsky, Gail Gazelle, Olga Saynina, and Arlene S. Ash. Chemotherapy use among medicare beneficiaries at the end of life. *Annals of Internal Medicine*, 138(8):639–643.

[45] Apostolos G. Fertis. *A robust optimization approach to statistical estimation problems*. PhD thesis, Massachusetts Institute of Technology, 2009.

[46] National Quality Forum. Cancer measures.

[47] Robert S. Franco. Measurement of red cell lifespan and aging. *Transfusion medicine and hemotherapy*, 39(5):302–307, 2012.

[48] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

[49] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[50] Zoubin Ghahramani and Michael I. Jordan. Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems*, pages 120–127, 1994.

[51] Paul Glare, Kiran Virik, Mark Jones, Malcolm Hudson, Steffen Eychmuller, John Simes, and Nicholas Christakis. A systematic review of physicians' survival predictions in terminally ill cancer patients. *BMJ*, 327(7408):195.

[52] Joseph A. Greer, William F. Pirl, Vicki A. Jackson, Alona Muzikansky, Inga T. Lennes, Rebecca S. Heist, Emily R. Gallagher, and Jennifer S. Temel. Effect of early palliative care on chemotherapy use and end-of-life care in patients with metastatic non-small-cell lung cancer. *Journal of Clinical Oncology*, 30(4):394–400.

[53] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual, 2016.

[54] Patrick L. Harrington, Jr., Aimee Zaas, Christopher W. Woods, Geoffrey S. Ginsburg, Lawrence Carin, and Alfred O. Hero, III. Robust Logistic Regression with Bounded Data Uncertainties. Technical report, University of Michigan, 09 2010.

[55] Takashi Hirose, Toshimitsu Yamaoka, Tsukasa Ohnishi, Tomohide Sugiyama, Sojiro Kusumoto, Takao Shirai, Kentaro Okuda, Tohru Ohmori, and Mitsuru Adachi. Patient willingness to undergo chemotherapy and thoracic radiotherapy for locally advanced non-small cell lung cancer. *Psycho-Oncology*, 18(5):483–489.

[56] James Honaker, Gary King, Matthew Blackwell, et al. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47, 2011.

[57] Peter J. Huber. *Robust statistics*. Wiley, New York, 1981.

[58] Guido W. Imbens and Donald B. Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

[59] S. E. Inzucchi, R. M. Bergenstal, J. B. Buse, M. Diamant, E. Ferrannini, M. Nauck, A. L. Peters, A. Tsapas, R. Wender, and D. R. Matthews. Management of hyperglycemia in type 2 diabetes: A patient-centered approach position statement of the American Diabetes Association (ADA) and the European Association for the Study of Diabetes (EASD). *Diabetes Care*, 35(6):1364–1379, 2012.

[60] Faramarz Ismail-Beigi, Etie Moghissi, Margaret Tiktin, Irl B. Hirsch, Silvio E. Inzucchi, and Saul Genuth. Individualizing glycemic targets in type 2 diabetes mellitus: Implications of recent clinical trials. *Annals of Internal Medicine*, 154(8):554–559, 2011.

[61] Michael I. Jordan and Tom M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[62] C. S. Karapetis, S. Khambata-Ford, D. J. Jonker, C. J. O'callaghan, D. Tu, N. C. Tebbutt, R. J. Simes, H. Chalchal, J. D. Shapiro, S. Robitaille, et al. K-ras mutations and benefit from cetuximab in advanced colorectal cancer. *New England Journal of Medicine*, 359(17):1757–1765, 2008.

[63] Mohammad E. Khan, Guillaume Bouchard, Kevin P. Murphy, and Benjamin M. Marlin. Variational bounds for mixed-data factor analysis. In *Advances in Neural Information Processing Systems*, pages 1108–1116, 2010.

[64] Hyunsoo Kim, Gene H. Golub, and Haesun Park. Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics*, 21(2):187–198, 2005.

[65] Ki-Yeol Kim, Byoung-Jin Kim, and Gwan-Su Yi. Reuse of imputed data in microarray analysis increases imputation efficiency. *BMC Bioinformatics*, 5(1):1, 2004.

[66] Moshe Lichman. UCI machine learning repository, 2013.

[67] Alan Wee-Chung Liew, Ngai-Fong Law, and Hong Yan. Missing value imputation for gene expression data: computational techniques to recover missing data from available information. *Briefings in Bioinformatics*, 12(5):498–513, 2011.

[68] Kasia J. Lipska, Clifford J. Bailey, and Silvio E. Inzucchi. Use of metformin in the setting of mild-to-moderate renal insufficiency. *Diabetes care*, 34(6):1431–1437, 2011.

[69] Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 1987.

[70] Roi Livni, Koby Crammer, Amir Globerson, ELSC-ICNC Edmond, and Lily Safra. A simple geometric interpretation of svm using stochastic adversaries. In *AISTATS*, pages 722–730, 2012.

[71] Miles Lubin and Iain Dunning. Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

[72] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11(Aug):2287–2322, 2010.

[73] Shakir Mohamed, Zoubin Ghahramani, and Katherine A. Heller. Bayesian exponential family PCA. In *Advances in Neural Information Processing Systems*, pages 1089–1096, 2009.

[74] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep K. Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems*, pages 1196–1204, 2013.

[75] National Center for Chronic Disease Prevention and Health Promotion. National Diabetes Statistics Report. Technical report, Centers for Disease Control and Prevention, 2014.

[76] Shigeyuki Oba, Masa-aki Sato, Ichiro Takemasa, Morito Monden, Ken-ichi Matsubara, and Shin Ishii. A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

[77] Raghav Pant, Theodore B. Trafalis, and Kash Barker. Support vector machine classification of uncertain and imbalanced data using robust optimization. In *Proceedings of the 15th WSEAS international conference on computers*, pages 369–374, 2011.

[78] Trivellore E. Raghunathan, James M. Lepkowski, John Van Hoewyk, and Peter Solenberger. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, 27(1):85–96, 2001.

[79] Wullianallur Raghupathi and Viju Raghupathi. Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2(1):3, 2014.

[80] H. W. Rodbard, P. S. Jellinger, J. A. Davidson, D. Einhorn, A. J. Garber, G. Grunberger, Y. Handelsman, E. S. Horton, H. Lebovitz, P. Levy, et al. Statement by an American Association of Clinical Endocrinologists/American College of Endocrinology consensus panel on type 2 diabetes mellitus: An algorithm for glycemic control. *Endocrine Practice*, 15(6):540–559, 2009.

[81] Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, pages 41–55, 1983.

[82] Lowell E. Schnipper, Thomas J. Smith, Derek Raghavan, Douglas W. Blayney, Patricia A. Ganz, Therese Marie Mulvey, and Dana S. Wollins. American society of clinical oncology identifies five key opportunities to improve care and reduce costs: The top five list for oncology. *Journal of Clinical Oncology*, 30(14):1715–1724.

[83] Nicholas J. Schork. Personalized medicine: time for one-person trials. *Nature*, 520(7549):609–611, 2015.

[84] Gerard Silvestri, Robert Pritchard, and H. Gilbert Welch. Preferences for chemotherapy in patients with advanced non-small cell lung cancer: descriptive study based on scripted interviews. *BMJ (Clinical research ed.)*, 317(7161):771–775.

[85] Daniel J. Stekhoven and Peter Bühlmann. Missforest: non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

[86] P. C. Stone and S. Lund. Predicting prognosis in patients with advanced cancer. *Annals of Oncology*, 18(6):971–976.

[87] Savitha Subramanian and Irl B. Hirsch. Personalized diabetes management: moving from algorithmic to individualized therapy. *Diabetes Spectrum*, 27(2):87–91, 2014.

[88] R Core Team. R: A language and environment for statistical computing, 2016.

[89] Terry Therneau, Beth Atkinson, and Brian Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015. R package version 4.1-9.

[90] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[91] Theodore B. Trafalis and Robin C. Gilbert. Robust support vector machines for classification and computational issues. *Optimisation Methods and Software*, 22(1):187–198, 2007.

[92] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[93] Stef Van Buuren. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, 16(3):219–242, 2007.

[94] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[95] Kiri Wagstaff. Clustering with missing values: No imputation required. In *Classification, Clustering, and Data Mining Applications*, pages 649–658. Springer, 2004.

[96] Xian Wang, Ao Li, Zhaohui Jiang, and Huanqing Feng. Missing value estimation for DNA microarray gene expression data by support vector regression imputation and orthogonal coding scheme. *BMC Bioinformatics*, 7(1):1, 2006.

[97] Alexi A. Wright, Baohui Zhang, Nancy L. Keating, Jane C. Weeks, and Holly G. Prigerson. Associations between palliative chemotherapy and adult cancer patients end of life care and place of death: prospective cohort study. *BMJ*, 348:g1219.

[98] Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[99] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510, 2009.

[100] C. Zhang and R. Zhang. More effective glycaemic control by metformin in African Americans than in Whites in the prediabetic population. *Diabetes & metabolism*, 41(2):173–175, 2015.

[101] Xiaobai Zhang, Xiaofeng Song, Huinan Wang, and Huanping Zhang. Sequential local least squares imputation estimating missing value of microarray data. *Computers in Biology and Medicine*, 38(10):1112–1120, 2008.

[102] Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. *Advances in neural information processing systems*, 16(1):49–56, 2004.

[103] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.