Machine Learning for Problems with Missing and Uncertain Data with Applications to Personalized Medicine

by

Colin Pawlowski

B.S., Yale University (2014)

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 2019

© Massachusetts Institute of Technology 2019. All rights reserved.
Abstract

When we try to apply statistical learning in real-world applications, we frequently encounter data which include missing and uncertain values. This thesis explores the problem of learning from missing and uncertain data with a focus on applications in personalized medicine.

In the first chapter, we present a framework for classification when data is uncertain that is based upon robust optimization. We show that adding robustness in both the features and labels results in tractable optimization problems for three widely used classification methods: support vector machines, logistic regression, and decision trees. Through experiments on 75 benchmark data sets, we characterize the learning tasks for which adding robustness provides the most value.

In the second chapter, we develop a family of methods for missing data imputation based upon predictive methods and formal optimization. We present formulations for models based on K-nearest neighbors, support vector machines, and decision trees, and we develop an algorithm OptImpute to find high quality solutions which scales to large data sets. In experiments on 84 benchmark data sets, we show that OptImpute outperforms state-of-the-art methods in both imputation accuracy and performance on downstream tasks.

In the third chapter, we develop MedImpute, an extension of OptImpute specialized for imputing missing values in multivariate panel data. This method is tailored for data sets that have multiple observations of the same individual at different points in time. In experiments on the Framingham Heart Study and Dana Farber Cancer Institute electronic health record data, we demonstrate that MedImpute improves the accuracy of models predicting 10-year risk of stroke and 60-day risk of mortality for late-stage cancer patients.

In the fourth chapter, we develop a method for tensor completion which leverages noisy side information available on the rows and/or columns of the tensor. We apply this method to the task of predicting anti-cancer drug response at particular dosages. We demonstrate significant gains in out-of-sample accuracy filling in missing values on two large-scale anti-cancer drug screening data sets with genomic side information.
Thesis Supervisor: Dimitris Bertsimas
Title: Boeing Leaders for Global Operations Professor
Co-Director, Operations Research Center
Acknowledgments

First and foremost, I would like to thank my thesis advisor, Dimitris Bertsimas, who has been a great friend and mentor to me during this PhD. Dimitris has helped me to grow and mature as a researcher, and he has inspired me to strive to make a big impact upon the world. In our research meetings, he always brings a strong dose of optimism and also a sense of humor which makes research fun and enjoyable. Thank you so much for guiding and supporting me over the past several years, and I will be forever grateful for everything that you have taught me.

I want to thank Rahul Mazumder and Colin Fogarty for generously agreeing to serve on my thesis committee. I have appreciated our interesting and insightful discussions about my research, and I have really enjoyed the seminar courses that I have taken with both of you which have given me inspiration for new research directions. In addition, thank you to Roy Welsch for serving on my general exam committee and providing feedback on my research in its early stages. Many other professors and instructors have had a significant influence upon me during my time at the ORC. In particular, thank you to Robert Freund, David Gamarnik, John Silberholz, and Martin Copenhaver for being great role models to me as teachers in the classroom.

Next, I would like to acknowledge multiple past and current students from the ORC who have contributed greatly to the chapters in this thesis and outside research projects. I cannot thank you enough for your contributions, and I will miss working with you day-to-day. Daisy Zhuo, thank you for being a close friend and colleague for many years. You have been an amazing collaborator, and I have enjoyed working with you on everything from OptImpute to hackathons to ORC t-shirt logos. Jack Dunn, thank you for your countless software contributions, your amazing writing, the occasional Australia joke, and always believing in our research team. Agni Orfanoudaki, thank you for being a close friend and collaborator especially over this past year. You are a truly exceptional researcher and academic thought leader, and it has been my privilege to work on MedImpute with you. Yuchen Wang, thank you for being a great partner on every industry project that I have been on, and also an excellent cubicle neighbor and lunch buddy. I have really enjoyed your sense of humor. Holly
Wiberg, thank you for your significant contributions in all of our healthcare collaborations. Your truly positive personality makes it a joy to do research with you, and I appreciate your taste in country music. Galit Lukin, thank you for being a great collaborator in the Quest project, which has been a huge help during a hectic final year. Finally, I want to thank Emma Gibson for our discussions on dose response curves which helped lay the groundwork for the final chapter of this thesis.

In addition to my research collaborators, I would also like to thank the rest of the students in my cohort and from other years who have helped me along this journey and made this time spent at the ORC special. Yee Sian Ng, thank you for being an amazing friend and supportive person since the very start of the program. Mathieu Dahan, thank you for being a great friend and one of the nicest people in the world. Andrew Li and Elisabeth Paulson, thank you for being excellent Muddy Charles co-captains, and special thanks to Andrew for giving me the best relationship advice of my life. Sebastien Martin, Max Burq, Max Biggs, Hugh Carson, and Spencer Wyant, thank you for hosting the best social events and making life at the ORC awesome. Nishanth Mundru, thank you for being a trusted friend and tea time buddy. Thank you to all of the friends that I have made at the ORC, including but not limited to: Joey Huchette, Jerry Kung, Velibor Mišić, Anna Papush, Charles Thraves, Amine Anoun, Rim Hariss, Michael Hu, Konstantina Mellou, Ilias Zadik, Jackie Baek, Tamar Cohen, Ryan Cory-Wright, Emily Meigs, Ted Papalexopoulos, Nicholas Renegar, Matthew Sobiesk, Brad Sturt, and Jessica Zhu.

Outside of school, I would like to thank my friends and family who have supported me in Boston. In particular, thank you to my 318 Beacon St roommates: Walker Bristol, Eric Stephen, Shrestha Singh, and Carmen Altes; memories of our poster events are some of my fondest. Thank you to Eric Fishman for continuing to be my close friend and running buddy, and my other friends from college who have kept in touch including David Cruz, Max Birger, and Max Andersen. Thank you to the Pawlowski family in the Boston area and beyond. You have made me feel like Boston is my second hometown. Thank you to the Head family who has been supportive from afar, especially Brian Head who is a lifelong friend. Last but definitely not least, thank you to Sarah Roundy for being my closest friend and partner over the past three years. I am incredibly grateful for your love and support, especially during
this difficult final stretch. Our New Hampshire road trips, country music concerts, and the Big Air ski jump event are among the biggest highlights of my time spent at MIT.

Finally, thank you to my immediate family who has been with me every step of the way since the beginning, and who have provided me with unconditional love and support. Mom, thank you for always going above and beyond to help me and showing me how to work hard. Dad, thank you for being a calming influence and always believing in me. Megan, thank you for being an amazing sister, a reliable friend and confidant, and always sticking up for me. I dedicate this thesis to the three of you.
To Mom, Dad, and Megan,
who have always been there for me.
Contents

1 Introduction 15

1.1 Motivation .............................................. 15
1.2 Prior Work .............................................. 16
1.3 Overview of this Thesis .................................. 16

2 Robust Classification 25

2.1 Introduction .............................................. 26

2.2 Overview of Classification Methods ......................... 31

2.2.1 Soft-Margin Support Vector Machines ..................... 32
2.2.2 Logistic Regression ...................................... 33
2.2.3 Decision Trees and CART ................................ 34

2.3 Brief Overview of Robust Optimization ....................... 37

2.4 Robustness Against Uncertainty in Features .................. 39

2.4.1 Motivating Feature-Robustness ......................... 40
2.4.2 Soft-Margin Support Vector Machines ..................... 41
2.4.3 Logistic Regression ...................................... 43
2.4.4 Optimal Decision Trees ................................. 45

2.5 Robustness Against Uncertainty in Labels ..................... 46

2.5.1 Motivating Label-Robustness ............................. 46
2.5.2 Soft-Margin Support Vector Machines .............................................. 47
2.5.3 Logistic Regression ................................................................. 50
2.5.4 Optimal Decision Trees ........................................................... 52

2.6 Robustness in Both Features and Labels ............................................. 55
2.6.1 Soft-Margin Support Vector Machines ........................................... 56
2.6.2 Logistic Regression ..................................................................... 57
2.6.3 Optimal Decision Trees ............................................................... 57

2.7 Computational Experiments with Synthetic Data Sets ......................... 58
2.7.1 Experimental Setup ..................................................................... 58
2.7.2 Classification Methods ............................................................... 60
2.7.3 Results ....................................................................................... 61

2.8 Computational Experiments with Real-world Data Sets ....................... 62
2.8.1 Experimental Setup ..................................................................... 62
2.8.2 Classification Methods ............................................................... 63
2.8.3 Pairwise Comparisons ................................................................. 68
2.8.4 Predicting the Effectiveness of Robust Classification ......................... 69
2.8.5 Comparison with Regularized Methods ........................................... 74
2.8.6 Computational Tractability and Speed ............................................. 78
2.8.7 The Price of Robustness ............................................................... 79

2.9 Conclusions .................................................................................... 81

3 Optimal Missing Data Imputation ........................................................... 83
3.1 Introduction ..................................................................................... 84
3.1.1 Related Work ............................................................................ 85
3.1.2 Contributions ............................................................................ 88
3.2 Methods for Optimal Imputation ........................................................ 89
Chapter 1

Introduction

1.1 Motivation

When we try to apply statistical learning in real-world applications, we frequently encounter data which include missing and uncertain values. For example, electronic health records (EHRs) are a rich data source that can be used for personalized medicine, but these data sets are seldom completely and exactly known. Systemic and random errors from imprecise measurements, incomplete lab tests, and mistakes in data entry may bias the results of statistical models trained on these data sets. To make the problem even more difficult, we often do not know the underlying model which generated these data perturbations or missingness patterns.

There are few methods that are effective in dealing with missing and uncertain data, and most require us to make strong assumptions that the data and perturbations are drawn from well-known distributions defined by a small number of parameters. For example, when implementing the expectation-maximization (EM) algorithm for missing data imputation [49], we may assume that the continuous features in the data are drawn from Normal distributions and that the categorical features are drawn from Multinomial distributions. When these assumptions are not met, then the imputations may be significantly inaccurate. Furthermore, if we try to fix this problem by cross-validating over more complicated distributions, the solution method quickly becomes intractable for large data sets.
This thesis focuses on machine learning methods for real-world problems with missing and uncertain data. In order to avoid distributional assumptions, we consider nonparametric models based upon state-of-the-art machine learning methods. We leverage insights from optimization to improve upon existing statistical methods for handling missing and uncertain data. In particular, we focus on a few concrete applications in personalized medicine.

1.2 Prior Work

Machine learning is a large, rapidly advancing field so there has been much progress in this area in recent decades. For example, different models for linear regression have been explored which mitigate the impact of uncertain data by adding regularizer terms \[142, 55\] or sparsity constraints \[17, 25\] to the least-squares formulation. Recently, several methods for matrix completion given noisy side information have been proposed which extend traditional matrix completion methods \[15, 59\]. Numerous methods for missing value imputation have been proposed in the last decade \[92, 33, 113, 81\]. Because this body of past work is quite extensive, we defer a more in-depth discussion of these advances and related works to the literature reviews in Chapters 2, 3, 4, and 5.

1.3 Overview of this Thesis

In this thesis, we propose methods based upon optimization to mitigate the negative impacts of uncertain and missing values on downstream learning tasks and apply these methods to applications in personalized medicine. For each method, we demonstrate tractability and out-of-sample performance gains on real-world data sets. We have grouped the major chapters of this thesis into two parts. In the first part, we develop methods which can be applied to general machine learning problems with missing and uncertain data. In the second part, we develop specialized methods which are tailored to applications in personalized medicine with missing and uncertain data. We describe each of the major chapters and their main contributions below. In Chapter 6, we provide concluding remarks and the major takeaways from this thesis.
Part 1: Machine Learning Methods for Missing and Uncertain Data

This part includes Chapters 2 and 3. Here, we develop methods that perform well on a broad range of benchmark data sets for machine learning.

Chapter 2 Robust Classification

In this chapter, we present a framework for classification when data is uncertain that is based upon robust optimization [14]. We show that adding robustness in both the features and labels results in tractable optimization problems for three widely used classification methods: support vector machines (SVM), logistic regression, and decision trees. Through experiments on 75 data sets from the University of California Irvine (UCI) machine learning repository, we show that all of the robust methods outperform their non-regularized nominal counterparts. In addition, we find an empirical decision rule that can be used to predict when adding robustness leads to the largest gains in out-of-sample accuracy over both regularized and non-regularized methods. The main contributions from this chapter are:

- We present a principled framework for classification that is robust to uncertainty in both the features and labels of the training data.

- We derive feature-robust, label-robust, and robust-in-both counterparts for SVM, logistic regression, and decision tree binary classification methods.

- We demonstrate the advantage of robust formulations in synthetic data experiments. In these experiments, we show that robust SVM and logistic regression methods recover separating hyperplane classifiers closer to the truth compared to regularized and nominal methods.

- We demonstrate the advantage of robust formulations in real-world data experiments. In particular, we show that adding robustness to SVM, logistic regression, and decision tree methods leads to gains in out-of-sample accuracy in the majority of experiments on 75 data sets from the UCI machine learning repository.

- We provide an empirically-derived decision rule that can be used to predict when
adding robustness provides the most value. In particular, the rule is:

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

where $p$ is the number of features of the data set and $a$ is the in-sample accuracy of the nominal classifier. When this rule is satisfied, then adding robustness leads to gains in out-of-sample accuracy of 5.3% for SVM, 2.1% for regularized SVM, 4.0% for logistic regression, 2.0% for regularized logistic regression, and 1.3% for decision trees across the UCI data sets.

Chapter 3 Optimal Missing Data Imputation

In this chapter, we develop a family of optimization-based imputation methods \texttt{opt.impute} which can be used to fill in missing values for general purpose applications [22]. We formulate the missing data problem with mixed continuous and categorical data as an optimization problem in which the imputed values are jointly optimized to minimize a global model-dependent cost function. We consider three different cost functions based upon different predictive models: $K$-nearest neighbors ($K$-NN), support vector machines (SVM), and decision trees. For each model, we derive an imputation method that finds high quality solutions to the corresponding optimization problem. In computational experiments on 84 data sets from the UCI machine learning repository, we demonstrate that \texttt{opt.impute} significantly outperforms state-of-the-art imputation methods in imputation accuracy. Further, we show how this framework can be extended to multiple imputation and leads to improved downstream performance compared to Multivariate Imputation via Chained Equations (MICE).

The main contributions from this chapter are:

- We develop a flexible optimization-based framework OptImpute for missing data imputation that accommodates a variety of imputation models based upon predictive models.

- We present formulations based upon three different predictive models: $K$-NN, SVM, and decision trees. These formulations can be used to impute data sets with mixed continuous and categorical features.
• For each formulation, 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 data sets with 100,000s of observations and 1,000s of features.

• We develop a cross-validated method `opt.cv` which selects the best imputation from the three predictive models $K$-NN, SVM, and decision trees.

• We develop a multiple imputation method `opt.mi` which uses the results of the `opt.impute` algorithm from different warm starts to generate multiple imputations.

• We run computational experiments to evaluate the performance of OptImpute relative to state-of-the-art methods. We compare against methods for single imputation including: Mean impute, $K$-NN, Iterative $K$-NN, Bayesian PCA, and Predictive-Mean Matching, and we develop a cross-validated benchmark which selects the best imputation among all of these methods. On downstream tasks, we also compare against MICE for multiple imputation.

• We demonstrate that `opt.cv` produces more accurate imputations than existing methods for single imputation. In particular, `opt.cv` decreases the average imputation error by 8.3\% compared to the best cross-validated benchmark in experiments on 84 data sets from the UCI machine learning repository.

• We demonstrate that the improved imputations from `opt.cv` result in improved performance on downstream learning tasks. On 10 classification and regression tasks with 50\% missing data, `opt.cv` improves the average classification accuracy from 84.4\% to 86.1\% and the average regression $R^2$ from 0.315 to 0.339 compared to best cross-validated benchmark method.

• We show that downstream models trained on multiple imputations produced by `opt.mi` significantly outperform multiple imputations produced by MICE in 3/5 missing data scenarios for classification and 5/5 scenarios for regression.
This part includes Chapters 4 and 5. In these chapters, we develop specialized methods for personalized medicine applications with missing and uncertain data.

**Chapter 4: Imputation of Clinical Covariates in Time Series**

In this chapter, we develop `med.impute`, an extension of `opt.impute` that is specialized for filling in missing values of clinical covariates in multivariate panel data [20]. This method is tailored for data sets that have multiple observations of the same individual at different points in time, which includes rich data sets such as longitudinal clinical studies, insurance claims, and electronic health records (EHRs). We consider an optimization formulation that leverages time series information by adding an exponential-decay regularizer term to the objective function of the `opt.impute` formulation. For the $K$-nearest neighbors ($K$-NN) model, we present the formulation and a fast first-order method to find high-quality solutions. Through computational experiments on two real-world clinical data sets, we demonstrate that `med.impute` gives significant improvements in both imputation accuracy and performance on downstream tasks compared to state-of-the-art methods. In particular, on longitudinal data from the Framingham Heart Study, this imputation method improves the accuracy of models predicting 10-year risk of stroke. On electronic health record from the Dana Farber Cancer institute, this method improves the accuracy of models predicting 60-day risk of mortality among late-stage cancer patients. The main contributions from this chapter are:

- We present an optimization-based framework MedImpute for imputing missing values in multivariate panel data, extending the OptImpute framework introduced in Chapter 3. This framework accommodates data sets that have multiple observations of the same individual observed at different points in time. Similar to OptImpute, MedImpute can be used to impute data sets with both continuous and categorical features, and we can formulate the optimization problem using a $K$-NN, SVM, or tree-based objective function.

- We present the MedImpute formulation under a $K$-NN objective function, and we derive a corresponding fast first-order algorithm `med.knn`.

- We run computational experiments comparing against state-of-the-art imputation
methods, including the original OptImpute method with a $K$-NN objective function $\text{opt.knn}$, along with Mean impute, Bayesian PCA, and MICE.

- We show that $\text{med.knn}$ outperforms the benchmark imputation methods on the Framingham Heart Study (FHS), a longitudinal data set with time series data recorded at regular time intervals. On the FHS data set with 50% data missing, $\text{med.knn}$ decreases the imputation error by 32.9% and improves the AUC of a logistic regression model for predicting 10-year risk of stroke from 0.773 to 0.837.

- We show that $\text{med.knn}$ outperforms the benchmark imputation methods on EHR data from the Dana Farber Cancer Institute, which is a less structured clinical data set that contains data recorded at irregular time intervals. On the Dana Farber EHR data set, $\text{med.knn}$ decreases the imputation error by 14.5% and improves the AUC of a logistic regression model for predicting 60-day risk of mortality among late-stage cancer patients from 0.896 to 0.901.

- We design and run informative experiments on the FHS and Dana Farber EHR data sets that evaluate the performance of $\text{med.knn}$ as we vary the percentage of missing data, number of observations per individual, and mechanism of missing data. We compare against the benchmark methods.

- We show that $\text{med.knn}$ consistently gives the highest AUC and lowest imputation error as we vary the missing percentage from 10% to 50%.

- We show that as we increase the number of observations per individual, $\text{med.knn}$ improves until it is the best method for both data sets.

- We show that $\text{med.knn}$ consistently gives the highest AUC and lowest imputation error as we vary the mechanism of missing data. In particular, we demonstrate that $\text{med.knn}$ works well on not missing at random (NMAR) missing patterns that are commonly encountered in practice for both longitudinal studies and EHR data.

Chapter 5: Tensor Completion with Noisy Side Information

In this chapter, we develop a new method for tensor completion with noisy side information,
and we apply this method to the problem of filling in missing values in anti-cancer drug response data sets [21]. This method is tailored for large-scale drug screening data sets that have drug response information across multiple patients, anti-cancer drugs, and doses, along with genomic information on the patients and drug target information on the anti-cancer drugs. Given (patient, drug, dose) data, we propose a tensor model which learns a low rank representation of the data and leverages information from the noisy side information as well. We also present several algorithms based upon alternating minimization to find high quality solutions which scale to large problem sizes. In computational experiments, we demonstrate that the tensor-based methods significantly outperform state-of-the-art methods for the prediction of drug response at specific doses on two large-scale anti-cancer drug screening data sets. In addition, the proposed method for tensor completion with noisy side information matches or outperforms the original tensor completion method in all of the experiments, and the relative improvement increases as the percentage of missing data increases. The main contributions from this chapter are:

- We propose a new model for tensor completion with noisy one-sided information. This model takes into account noisy features available on the rows of the tensor.

- We propose a new model for tensor completion with noisy two-sided information. This model takes into account noisy features available on both the rows and columns of the tensor.

- For each tensor model, we develop an alternating minimization algorithm to find high-quality solutions that is tractable for large problem sizes. We present the algorithm TensorOneSided to solve the one-sided information case, and the algorithm TensorTwoSided to solve the two-sided information case.

- We formulate the problem of predicting drug response as the problem of tensor completion with noisy side information. Under this framework, the three dimensions of the tensor are (patient, drug, dose), the row side information are genomic features of the patients, and the column side information are drug target and pathway features of the drugs.
• We run computational experiments on two large-scale anti-cancer drug screening data sets: the Genomics of Drug Sensitivity in Cancer (GDSC) and the Cancer Cell Line Encyclopedia (CCLE). We compare TensorGenomic against state-of-the-art methods including the original method for tensor completion without side information and a multi-level mixed effects model which was used to fit dose response curves in the original GDSC paper.

• We demonstrate that the proposed method TensorGenomic outperforms the state-of-the-art methods for predicting anti-cancer drug response on both the GDSC and CCLE data sets. In particular, with 80% missing data, TensorGenomic improves the $R^2$ from 0.404 to 0.552 in the GDSC data set and improves the $R^2$ from 0.407 to 0.524 in the CCLE data set compared to the best of the benchmark methods.
Chapter 2

Robust Classification

This work appeared in the INFORMS Journal on Optimization, with co-authors Dimitris Bertsimas, Jack Dunn, and Ying Daisy Zhuo [14].

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.
2.1 Introduction

Three of the most widely used classification methods are SVM (Support Vector Machines), logistic regression, and CART (Classification and Regression Trees) [56]. 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 work, 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 [46] 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 [134]. Here, 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 [55]. 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 [32] construct the partitions with greedy heuristic methods, although recently methods have been developed that efficiently find globally optimal solutions to the decision tree problem [12]. 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 [19] and Best Subset Selection problems [17], and to create algorithmic approaches for fitting regression models [15, 16]. 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 \((x_i, i = 1, \ldots, n)\) 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. 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 [6] 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 [10]. 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. [102] demonstrate how robust classification can be used to handle situations with imbalanced training data, and [85] derive classifiers protected against stochastic adversarial perturbations to the training data. [134] 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 [138, 27]. These approaches have dealt mainly with feature uncertainty. One of the robust classification methods proposed here, namely feature-robust SVM, closely resembles the linear optimization robust classifiers proposed by [121], 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 [134, 11]. 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 [28, 95]. There has also been work on robustifying kernel SVM methods against feature uncertainty by [5]. The approach we present could be extended to kernel methods, but this is beyond the scope of this work.

For logistic regression, regularized versions such as Elastic Net have been proposed [132], which consider adding a convex combination of the $\ell_1$ and $\ell_2$-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 [50, 62].

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 [12], 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 classification 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**
In this work, we show 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 im-
provement 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 [64], 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 this chapter is as follows. In Section 2.2, we present a selection of widely-used classification methods. In Section 2.3, we give a brief introduction to robust optimization and introduce some terms and properties that will be used later. In Section 2.4, 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 2.5, we repeat this process to develop methods that are robust to uncertainty in data set labels. In Section 2.6, we combine these approaches to develop classification methods that are robust to noise in both features and labels. In Section 2.7, 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 2.8, 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 2.9.

2.2 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,
let \( \{x_i, y_i\}_{i=1}^n \) be the training data provided for the classification task, where \( x_i \in \mathbb{R}^p \) is the feature vector and \( y_i \in \{-1, 1\} \) is the label for observation \( i \).

### 2.2.1 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 [46]. 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:

\[
\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max\{1 - y_i(w^T x_i - b), 0\}.
\] (2.1)

Problem (2.1) can equivalently be formulated as the following problem:

\[
\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i
\]
\[\text{s.t.} \quad y_i(w^T x_i - b) \geq 1 - \xi_i \quad \forall i, \]
\[\xi_i \geq 0 \quad \forall i.
\] (2.2)

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

\[
\max_\alpha \quad C \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j
\]
\[\text{s.t.} \quad 0 \leq \alpha_i \leq C \quad \forall i, \]
\[\sum_{i=1}^n \alpha_i y_i = 0.
\]

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 [56]. In addition, the dual form is advantageous because it allows us to do the
kernel trick to learn non-linear decision rules [46]. Alternatively, we may modify the objective function of problem (2.1) by changing the norm of the regularizer term from $\ell_2$ to $\ell_1$ [141]. The resulting classifier is formulated as follows:

$$
\min_{w, b} \|w\|_1 + C \sum_{i=1}^{n} \xi_i \\
\text{s.t. } y_i (w^T x_i - b) \geq 1 - \xi_i \quad \forall i,
$$

$$
\xi_i \geq 0 \quad \forall i. \tag{2.3}
$$

Problem (2.3), which we refer to as $\ell_1$-regularized SVM, is equivalent to a linear optimization problem which is efficiently solvable.

### 2.2.2 Logistic Regression

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

$$
P(Y = 1|X = x) = \frac{e^{\beta^T x + \beta_0}}{1 + e^{\beta^T x + \beta_0}},
$$

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

Concisely, the conditional probability can be written as

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

Logistic regression coefficients $\beta$ and $\beta_0$ are typically fit using maximum likelihood method. The log-likelihood is

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

Therefore, the maximum-likelihood estimators $\beta$ and $\beta_0$ aim to solve the following problem:

$$
\max_{\beta, \beta_0} - \sum_{i=1}^{n} \log \left( 1 + e^{-y_i(\beta^T x_i + \beta_0)} \right). \tag{2.4}
$$
Problem (2.4) is a concave maximization problem that is efficiently solvable by methods such as coordinate descent or Newton’s method \[9\].

Similar to the regularization techniques in the popular lasso regression \[119\] for variable selection and shrinkage, a regularization term can be added to the logistic regression likelihood function, giving

\[
\max_{\beta, \beta_0} - \sum_{i=1}^{n} \log \left( 1 + e^{-y_i (\beta^T x_i + \beta_0)} \right) - \lambda \| \beta \|_q,
\]

where \( \| \cdot \|_q \) is a given \( \ell_q \) norm.

### 2.2.3 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 \[32\], 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 \[12\] 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 \( a^T x < b \) 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 2-1 summarizes this for an example decision tree with two branch nodes, A and B, that apply splits \( a_A^T x < b_A \) and \( a_B^T x < b_B \) 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 $a_k^T x_i < b_k$) 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 $a_k^T x_i \geq b_k$) to get to node $k$;
- $\mathcal{P}_k = \mathcal{P}_k^L \cup \mathcal{P}_k^R$, i.e., all ancestors of node $k$.

We will now state the Optimal Decision Tree problem from \[12\] below as Problem (2.6) and then provide an explanation of the model:

\[
\begin{align*}
\min & \quad \sum_{k=1}^{K} f_k - \sum_{k=1}^{K} \lambda_k d_k \\
\text{s.t.} & \quad g_k = \sum_{i=1}^{n} \frac{1 - y_i}{2} z_{ik} \quad \forall k, \\
& \quad h_k = \sum_{i=1}^{n} \frac{1 + y_i}{2} z_{ik} \quad \forall k, \\
& \quad f_k \leq g_k + M[w_k + (1 - c_k)] \quad \forall k, \\
& \quad f_k \leq h_k + M[(1 - w_k) + (1 - c_k)] \quad \forall k, \\
& \quad f_k \geq g_k - M[(1 - w_k) + (1 - c_k)] \quad \forall k, \\
& \quad f_k \geq h_k - M[w_k + (1 - c_k)] \quad \forall k,
\end{align*}
\]
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 $a_k$ and $b_k$ are used to set a split of the form $a_k^T x < b_k$. 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 (2.6r), which forces the components of $a_k$ to be binary, and (2.6j) means we can only choose one of these variables at each node. Note that (2.6j) also forces $a = 0$ 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 (2.6p) and (2.6q) ensure that points are assigned only
to a node if they satisfy all required splits, while constraints (2.6l) and (2.6m) ensure that points can only be assigned to leaf nodes. Finally, (2.6k) 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 (2.6b) and (2.6c). Constraints (2.6d) through (2.6g) set $f_k$ to $\text{min}\{g_k, h_k\}$ 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 (2.6n) and (2.6o) 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 [12] 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 [12] for a comprehensive comparison).

2.3 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 with-
out 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:

$$\max_{x \in \mathcal{X}} \ c(u, x)$$

s.t. $g(u, x) \leq 0$,

where $x$ is the vector of decision variables, $u$ is a vector of given parameters, $c$ is a real-valued function, $g$ is a vector-valued function, and $0$ is the vector of all zeros. Relaxing the assumption that $u$ is fixed, we assume instead that the realized values of $u$ 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:

$$\max_{x \in \mathcal{X}} \ \min_{u \in \mathcal{U}} \ c(u, x)$$

s.t. $g(u, x) \leq 0 \ \forall u \in \mathcal{U}$.

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 [7].
Dual Norms

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

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

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

\[
\max_{\|x\|_q \leq 1} \{a^T x\}.
\]

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

\[
\|x\|_\infty = \lim_{q \to \infty} \|x\|_q = \max\{|x_1|, |x_2|, \ldots, |x_n|\}.
\]

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

\[
\max_{\|x\|_q \leq \rho} \{a^T x\} = \max_{\|y\|_{q^*} \leq 1} \{a^T (\rho y)\} = \rho \cdot \max_{\|y\|_q \leq 1} \{a^T y\},
\]

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

2.4 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.
2.4.1 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 $x_i$ are subject to additive perturbations $\Delta x_i \in \mathbb{R}^p, i = 1, \ldots, n$. Let $\Delta X = (\Delta x_1, \Delta x_2, \ldots, \Delta x_n)$ and define the following uncertainty set:

$$\mathcal{U}_x = \{\Delta X \in \mathbb{R}^{n \times p} \mid \|\Delta x_i\|_q \leq \rho, i = 1, \ldots, n\}, \quad (2.8)$$

where $\rho$ 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 $x_i + \Delta x_i, i = 1, \ldots, n$. We now seek to construct a classifier that is robust to all such perturbations $\Delta X \in \mathcal{U}_x$. To do this, we robustify against this uncertainty set of feature parameters in each of our classification methods. In practice, the parameter $\rho$ 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 $\{\Delta x_i \in \mathbb{R}^p \mid \|\Delta x_i\|_q \leq \rho\}; i = 1, \ldots, n$. 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 con-
sider the norm uncertainty set $\mathcal{U}_x$ because it admits a simple geometric interpretation and only requires tuning a single parameter $\rho$, 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.

### 2.4.2 Soft-Margin Support Vector Machines

The regularized Support Vector Machine problem in (2.2) has been shown by [134] and [52] 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 $\|w\|^2_2$ is a by-product of feature robustness. Further discussion of the equivalence between classical SVM and feature-robust formulations is provided in Appendix A.1. 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:

$$\min_{w, b} \sum_{i=1}^{n} \max \{1 - y_i (w^T x_i - b), 0\}. \quad (2.9)$$

Robustifying Problem (2.9) against the uncertainty set $\mathcal{U}_x$ gives the following robust optimization problem:

$$\min_{w, b} \max_{\Delta x \in \mathcal{U}_x} \sum_{i=1}^{n} \max \{1 - y_i (w^T (x_i + \Delta x_i) - b), 0\}. \quad (2.10)$$

We now derive the robust counterpart to Problem (2.10). Note that this is equivalent to Theorem 3 in [134].
Theorem 1. The robust counterpart to Problem (2.10) is

\[
\min_{w,b} \sum_{i=1}^{n} \xi_i \quad \text{s.t. } y_i(w^T x_i - b) - \rho \|w\|_q^* \geq 1 - \xi_i \quad \forall i, \\
\xi_i \geq 0 \quad \forall i, 
\]

(2.11)

where \(\ell_q^*\) is the dual norm of \(\ell_q\).

Proof. We can reformulate Problem (2.10) as

\[
\min_{w,b} \sum_{i=1}^{n} \xi_i \quad \text{s.t. } y_i(w^T(x_i + \Delta x_i) - b) \geq 1 - \xi_i \quad \forall \Delta X \in \mathcal{U}_x \quad \forall i, \\
\xi_i \geq 0 \quad \forall i. 
\]

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

\[
\min_{\Delta X \in \mathcal{U}_x} (y_i w^T \Delta x_i) \geq 1 - \xi_i - y_i (w^T x_i - b) \quad \forall i. 
\]

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

\[
\min_{\Delta x_i} y_i w^T \Delta x_i \quad \text{s.t. } \|\Delta x_i\|_q \leq \rho. 
\]

Because \(y_i\) is constant, we recognize this optimization problem as the dual norm problem. Therefore, by (2.7), for any given value of \(w\), the objective value of this problem is \(-\rho \|w\|_q^*\), where \(\ell_q^*\) is the dual norm of \(\ell_q\). Replacing the minimization term with this optimal value and rearranging yields (2.11).

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 [9]. When \(q = 1\), \(q^* = \infty\)
or \( q = \infty, \ q^* = 1 \), feature-robust SVM can be reformulated as a linear optimization problem.

### 2.4.3 Logistic Regression

Robustifying Problem (2.4) against the uncertainty set \( \mathcal{U}_x \) yields the following robust optimization problem:

\[
\max_{\beta, \beta_0} \min_{\Delta x \in \mathcal{U}_x} \sum_{i=1}^{n} \log \left( 1 + e^{-y_i(\beta^T (x_i + \Delta x_i) + \beta_0)} \right).
\]  

(2.12)

Next we determine the robust counterpart to Problem (2.12). We note that similar results on more specific uncertainty sets have been previously shown in [50, 62].

**Theorem 2.** The robust counterpart to Problem (2.12) is

\[
\max_{\beta, \beta_0} \sum_{i=1}^{n} \log \left( 1 + e^{-y_i(\beta^T x_i + \beta_0)} + \rho \| \beta \| q^* \right),
\]  

(2.13)

where \( \ell_{q^*} \) is the dual norm of \( \ell_q \).

**Proof.** Consider the inner minimization problem in (2.12), which is the following optimization problem:

\[
\min_{\Delta x \in \mathcal{U}_x} \sum_{i=1}^{n} \log \left( 1 + e^{-y_i(\beta^T (x_i + \Delta x_i) + \beta_0)} \right).
\]  

(2.14)

Let \( \omega_i = y_i(\beta^T (x_i + \Delta x_i) + \beta_0) \), and define \( g(\omega_i) = -\log (1 + e^{-\omega_i}) \). The first-order derivative of \( g \) with respect to \( \omega_i \) 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 (2.12) is the same as the solution of the problem

\[
\min_{\Delta x_i} y_i(\beta^T (x_i + \Delta x_i) + \beta_0)
\]  

s.t. \( \| \Delta x_i \|_q \leq \rho \).  

(2.15)
This is equivalent to the following problem:

\[ y_i(\beta^T x_i + \beta_0) - \max_{\Delta x_i} - y_i \beta^T \Delta x_i \]

s.t. \( \|\Delta x_i\|_q \leq \rho. \)

We recognize this maximization term as the dual norm problem. Therefore, by (2.7), the optimal solution is \( \rho \|\beta\|_{\ell^q}, \) where \( \ell^q \) is the dual norm of \( \ell_q. \) We conclude that the optimal value to (2.15) is \( y_i(\beta^T x_i + \beta_0) - \rho \|\beta\|_{\ell^q}. \) Substituting the optimal value into the inner minimization problem in (2.12), we obtain

\[-\sum_{i=1}^n \log \left( 1 + e^{-y_i(\beta^T x_i + \beta_0) + \rho \|\beta\|_{\ell^q}} \right).\]

Maximizing the above equation over \( \beta, \beta_0 \) yields (2.13). \( \square \)

If \( q \geq 2, \) the robust counterpart (2.13) 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 (2.13) 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 [9].

Compared to the nominal case, the feature-robust counterpart of logistic regression has an additional \( \rho \|\beta\|_{\ell^q} \) term in the exponent of the logit function. It resembles the regularization term in regularized logistic regression, shown in Equation (2.5). 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:

\[-\sum_{i=1}^n \log \left( 1 + e^{-y_i(\beta^T x_i + \beta_0)} \right) - \sum_{i=1}^n \frac{e^{-y_i(\beta^T x_i + \beta_0)}}{1 + e^{-y_i(\beta^T x_i + \beta_0)}} \rho \|\beta\|_{\ell^q}.\]

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

### 2.4.4 Optimal Decision Trees

Robustifying Problem (2.6) against the uncertainty set \( \mathcal{U}_x \) gives a problem identical to Problem (2.6) except with the following constraints in place of the constraints (2.6p) and (2.6q):

\[
\begin{align*}
\mathbf{a}_j^T (\mathbf{x}_i + \Delta \mathbf{x}_i) + \epsilon &\leq b_j + M (1 - z_{ik}) \quad \forall \Delta \mathbf{X} \in \mathcal{U}_x, \quad \forall i, k, \forall j \in \mathcal{P}^l_k, \quad (2.16a) \\
\mathbf{a}_j^T (\mathbf{x}_i + \Delta \mathbf{x}_i) &\geq b_j + M (1 - z_{ik}) \quad \forall \Delta \mathbf{X} \in \mathcal{U}_x, \quad \forall i, k, \forall j \in \mathcal{P}^u_k. \quad (2.16b)
\end{align*}
\]

We refer to this optimization problem as Problem (2.16).

**Theorem 3.** The robust counterpart to Problem (2.16) is identical to Problem (2.16) except with the following constraints in place of constraints (2.16a) and (2.16b):

\[
\begin{align*}
\mathbf{a}_j^T \mathbf{x}_i + \rho + \epsilon &\leq b_j + (1 - z_{ik}) \quad \forall i, k, \forall j \in \mathcal{P}^l_k, \quad (2.17a) \\
\mathbf{a}_j^T \mathbf{x}_i - \rho &\geq b_j + (1 - z_{ik}) \quad \forall i, k, \forall j \in \mathcal{P}^l_k. \quad (2.17b)
\end{align*}
\]

**Proof.** Because constraint (2.16a) must hold for all \( \Delta \mathbf{X} \in \mathcal{U}_x \), this constraint is equivalent to

\[
\max_{\Delta \mathbf{X} \in \mathcal{U}_x} \{ \mathbf{a}_j^T \Delta \mathbf{x}_i \} \leq b_j + M (1 - z_{ik}) - \mathbf{a}_j^T \mathbf{x}_i - \epsilon \quad \forall i, k, \forall j \in \mathcal{P}^l_k.
\]

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

\[
\max \mathbf{a}_j^T \Delta \mathbf{x}_i \\
\text{s.t.} \quad \|\Delta \mathbf{x}_i\|_q \leq \rho.
\]

We recognize this as the dual norm problem, and by (2.7), the optimal value is \( \rho \|\mathbf{a}_j\|_{q^*} \), where \( \ell_{q^*} \) is the dual norm of \( \ell_q \). Moreover, if this constraint is to be non-trivial (which requires \( z_{ik} = 1 \)), we know from (2.6m) that \( d_j = 0 \) for all ancestors \( j \in \mathcal{P}^l_k \). Thus, from (2.6i) we have that \( \sum_i \mathbf{a}_{jl} = 1 \) and so together with (2.6r) we know that a single
element of $a_j$ is 1 with all other elements being 0. This means that $\|a_j\|_{q^*} = 1$ for any $q$, so the value of the maximization term is simply $\rho$. Rearranging terms yields the constraint (2.17a). We use an identical approach to yield (2.17b) from (2.16b).

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 $\rho$ around each $b_j$. The problem is therefore practically solvable like the nominal problem.

### 2.5 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 $\Gamma$. We then define and state the label-robust counterpart for each of the classification methods.

#### 2.5.1 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 been flipped, and 0 indicates that the label was correct. We consider the uncertainty set:

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

where $\Gamma$ 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 $\Delta y \in \mathcal{U}_y$ to obtain a classifier that is label-robust. In practice, the parameter $\Gamma$ 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 2.4, 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.

2.5.2 Soft-Margin Support Vector Machines

Robustifying Problem (2.2) against the uncertainty set $\mathcal{U}_y$ gives

$$\min_{w,b} \max_{\Delta y \in \mathcal{U}_y} \sum_{i=1}^{n} \max\{1 - y_i (1 - 2\Delta y_i) (w^T x_i - b), 0\}. \quad (2.18)$$
Theorem 4. The robust counterpart to Problem (2.18) is

$$\min \sum_{i=1}^{n} \xi_i + \Gamma q + \sum_{i=1}^{n} r_i$$

s.t. \(q + r_i \geq \phi_i - \xi_i\) \quad \forall i,

\(\xi_i \geq 1 - y_i(w^T x_i - b)\) \quad \forall i,

\(\xi_i \leq 1 - y_i(w^T x_i - b) + M(1 - s_i)\) \quad \forall i,

\(\xi_i \leq Ms_i\) \quad \forall i,

\(\phi_i \geq 1 + y_i(w^T x_i - b)\) \quad \forall i,

\(\phi_i \leq 1 + y_i(w^T x_i - b) + M(1 - t_i)\) \quad \forall i,

\(\phi_i \leq Mt_i\) \quad \forall i,

\(r_i, \xi_i, \phi_i \geq 0\) \quad \forall i,

\(q \geq 0\),

s.t \(t \in \{0, 1\}^n\).

where \(M\) is a sufficiently large constant.

Proof. Fix \(w\) and \(b\), and consider the inner maximization problem

$$\max_{\Delta y \in \mathcal{U}_y} \sum_{i=1}^{n} \max\{1 - y_i(1 - 2\Delta y_i)(w^T x_i - b), 0\} \quad \forall i. \quad (2.20)$$

Define the functions

$$f_i(\Delta y_i) = \max\{1 - y_i(1 - 2\Delta y_i)(w^T x_i - b), 0\}, \quad \forall i.$$ 

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

$$f_i(\Delta y_i) = [f_i(1) - f_i(0)]\Delta y_i + f_i(0)\quad \forall i.$$
Let $\phi_i = f_i(1)$ and $\xi_i = f_i(0)$ for $i = 1, \ldots, n$. It follows that Problem (2.20) is equivalent to

$$\max \sum_{i=1}^{n} (\phi_i - \xi_i) \Delta y_i + \xi_i$$

$$\text{s.t.} \quad \Delta y \in \mathcal{U}_y.$$

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

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

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

$$\max \sum_{i=1}^{n} (\phi_i - \xi_i) \Delta y_i + \xi_i$$

$$\text{s.t.} \quad 0 \leq \Delta y_i \leq 1 \quad \forall i,$$

$$\sum_{i=1}^{n} \Delta y_i \leq \Gamma.$$

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

$$\min \Gamma q + \sum_{i=1}^{n} r_i + \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad q + r_i \geq \phi_i - \xi_i \quad \forall i,$$

$$r_i \geq 0 \quad \forall i,$$

$$q \geq 0.$$
Minimizing over \( w \) and \( b \), this optimization problem becomes

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} \xi_i + \Gamma q + \sum_{i=1}^{n} r_i \\
\text{s.t.} & \quad q + r_i \geq \phi_i - \xi_i \quad \forall i, \\
& \quad \xi_i = \max\{1 - y_i(w^T x_i - b), 0\} \quad \forall i, \\
& \quad \phi_i = \max\{1 + y_i(w^T x_i - b), 0\} \quad \forall i, \\
& \quad r_i \geq 0 \quad \forall i, \\
& \quad q \geq 0.
\end{align*}
\]

Reformulating the problem to specify the values of the variables \( \xi_i, \phi_i \) with linear constraints yields the desired result.

Problem (2.19) is a mixed-integer optimization problem which is practically solvable.

### 2.5.3 Logistic Regression

Robustifying Problem (2.4) against the uncertainty set \( \mathcal{U}_y \) gives

\[
\begin{align*}
\max & \quad \beta, \beta_0 \\
\min_{\Delta y \in \mathcal{U}_y} & \quad \sum_{i=1}^{n} \log \left( 1 + e^{-(y_i(1 - 2\Delta y_i)(\beta^T x_i + \beta_0))} \right).
\end{align*}
\]

(2.21)

**Theorem 5.** The robust counterpart to Problem (2.21) is

\[
\begin{align*}
\max_{\beta, \beta_0} & \quad -\sum_{i=1}^{n} \log \left( 1 + e^{-y_i(\beta^T x_i + \beta_0)} \right) + \Gamma \mu + \sum_{i=1}^{n} \nu_i \\
\text{s.t.} & \quad \mu + \nu_i \leq \log \left( \frac{1 + e^{-y_i(\beta^T x_i + \beta_0)}}{1 + e^{y_i(\beta^T x_i + \beta_0)}} \right) \quad \forall i, \\
& \quad \nu_i \leq 0 \quad \forall i, \\
& \quad \mu \leq 0.
\end{align*}
\]

(2.22)

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

$$f_i(\Delta y_i) = [f(1) - f(0)]\Delta y_i + f(0) = \log \left( \frac{1 + e^{-y_i(\beta^T x_i + \beta_0)}}{1 + e^{y_i(\beta^T x_i + \beta_0)}} \right) \Delta y_i - \log \left( 1 + e^{y_i(\beta^T x_i + \beta_0)} \right).$$

We can thus rewrite the inner minimization part of Problem (2.21) as

$$\min_{\Delta y \in \mathcal{U}_y} \sum_{i=1}^n \left[ \log \left( \frac{1 + e^{-y_i(\beta^T x_i + \beta_0)}}{1 + e^{y_i(\beta^T x_i + \beta_0)}} \right) \Delta y_i - \log \left( 1 + e^{y_i(\beta^T x_i + \beta_0)} \right) \right]. \quad (2.23)$$

Since the convex hull of $\mathcal{U}_y$ has integer extreme points, Problem (2.23) has the same objective as its linear optimization relaxation [24]

$$\min_{\Delta y} \sum_{i=1}^n \left[ \log \left( \frac{1 + e^{-y_i(\beta^T x_i + \beta_0)}}{1 + e^{y_i(\beta^T x_i + \beta_0)}} \right) \Delta y_i - \log \left( 1 + e^{y_i(\beta^T x_i + \beta_0)} \right) \right] \quad \text{s.t.} \quad 0 \leq \Delta y_i \leq 1 \quad \forall i, \quad \sum_{i=1}^n \Delta y_i \leq \Gamma. \quad (2.24)$$

By strong duality, the optimal value to Problem (2.24) is equal to that of its dual problem

$$\max - \sum_{i=1}^n \log \left( \frac{1 + e^{-y_i(\beta^T x_i + \beta_0)}}{1 + e^{y_i(\beta^T x_i + \beta_0)}} \right) + \Gamma \mu + \sum_{i=1}^n \nu_i \quad \text{s.t.} \quad \mu + \nu_i \leq \log \left( \frac{1 + e^{-y_i(\beta^T x_i + \beta_0)}}{1 + e^{y_i(\beta^T x_i + \beta_0)}} \right) \quad \forall i, \quad \nu_i \leq 0 \quad \forall i, \quad \mu \leq 0.$$ 

Substituting this back into Problem (2.21) 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 con-
2.5.4 Optimal Decision Trees

Robustifying Problem (2.6) against the uncertainty set \( \mathcal{U}_y \) gives a problem identical to Problem (2.6) with the following constraints in place of constraints (2.6b), (2.6c), (2.6d), (2.6e), (2.6f), and (2.6g):

\[
g_k = \sum_{i=1}^{n} \frac{1 - y_i(1 - 2\Delta y_i)}{2} z_{ik} \quad \forall k, \quad (2.25a)
\]

\[
h_k = \sum_{i=1}^{n} \frac{1 + y_i(1 - 2\Delta y_i)}{2} z_{ik} \quad \forall k, \quad (2.25b)
\]

\[
f_k \leq g_k + M[w_k + (1 - c_k)] \quad \forall \Delta y \in \mathcal{U}_y, \quad \forall k, \quad (2.25c)
\]

\[
f_k \leq h_k + M[\{(1 - w_k) + (1 - c_k)] \quad \forall \Delta y \in \mathcal{U}_y, \quad \forall k, \quad (2.25d)
\]

\[
f_k \geq g_k - M[\{(1 - w_k) + (1 - c_k)] \quad \forall \Delta y \in \mathcal{U}_y, \quad \forall k, \quad (2.25e)
\]

\[
f_k \geq h_k - M[w_k + (1 - c_k)] \quad \forall \Delta y \in \mathcal{U}_y, \quad \forall k. \quad (2.25f)
\]

We refer to this optimization problem as Problem (2.25).

**Theorem 6.** The robust counterpart to Problem (2.25) is identical to Problem (2.25) with the following constraints in place of constraints (2.25a), (2.25b), (2.25c), (2.25d), (2.25e), and (2.25f):

\[
g_k = \sum_{i=1}^{n} \frac{1 - y_i}{2} z_{ik} \quad \forall k, \quad (2.26a)
\]

\[
h_k = \sum_{i=1}^{n} \frac{1 + y_i}{2} z_{ik} \quad \forall k, \quad (2.26b)
\]

\[
f_k \leq g_k - \Gamma \mu_{1,k} - \sum_{i=1}^{n} \nu_{1,ik} + M[w_k + (1 - c_k)] \quad \forall k, \quad (2.26c)
\]

\[
f_k \leq h_k - \Gamma \mu_{2,k} - \sum_{i=1}^{n} \nu_{2,ik} + M[\{(1 - w_k) + (1 - c_k)] \quad \forall k, \quad (2.26d)
\]

\[
f_k \geq g_k + \Gamma \mu_{3,ik} - \sum_{i=1}^{n} \nu_{3,ik} - M[\{(1 - w_k) + (1 - c_k)] \quad \forall k, \quad (2.26e)
\]
\[ f_k \geq h_k + \Gamma \mu_{4,k} + \sum_{i=1}^{n} \nu_{4,ik} - M[w_k + (1 - c_k)] \quad \forall k, \quad (2.26f) \]

\[ \mu_{m,k} + \nu_{m,ik} \geq -y_i z_{ik} \quad \forall i, k, \forall m \in \{1, 4\}, \quad (2.26g) \]

\[ \mu_{m,k} + \nu_{m,ik} \geq y_i z_{ik} \quad \forall i, k, \forall m \in \{2, 3\}, \quad (2.26h) \]

\[ \mu_{m,k}, \nu_{m,ik} \geq 0 \quad \forall i, k, m. \quad (2.26i) \]

**Proof.** We can substitute (2.25a) into constraint (2.25c) to obtain

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

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

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

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

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 \( k \))

\[
\min \sum_{i=1}^{n} y_i z_{ik} \Delta y_i
\]

s.t. \( 0 \leq \Delta y_i \leq 1 \quad \forall i, \)

\[
\sum_{i=1}^{n} \Delta y_i \leq \Gamma.
\]

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

\[
\max \Gamma \mu_{1,k} + \sum_{i=1}^{n} \nu_{1,ik}
\]

s.t. \( \mu_{1,k} + \nu_{1,ik} \leq y_i z_{ik} \quad \forall i, \)

\[
\mu_{1,k}, \nu_{1,ik} \leq 0 \quad \forall i.
\]
Substituting this back into the original constraint gives
\[
\sum_{i=1}^{n} \frac{1 - y_i}{2} z_{ik} + \Gamma \mu_{1,k} + \sum_{i=1}^{n} \nu_{1,ik} \geq f_k - M[w_k + (1 - c_k)] \quad \forall k,
\]
\[
\mu_{1,k} + \nu_{1,ik} \leq y_i z_{ik} \quad \forall i,
\]
\[
\mu_{1,k}, \nu_{1,ik} \leq 0 \quad \forall i.
\]

We substitute back for the original definition of \( g_k \) from (2.6b), and change the signs of \( \mu \) and \( \nu \) to get
\[
g_k - \Gamma \mu_{1,k} - \sum_{i=1}^{n} \nu_{1,ik} \geq f_k - M[w_k + (1 - c_k)] \quad \forall k,
\]
\[
\mu_{1,k} + \nu_{1,ik} \geq -y_i z_{ik} \quad \forall i,
\]
\[
\mu_{1,k}, \nu_{1,ik} \geq 0 \quad \forall i.
\]

We can rearrange this to obtain constraint (2.26c), as well as parts of constraints (2.26g) and (2.26i).

We repeat this entire process identically for constraints (2.25d), (2.25e), and (2.25f) 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 \( \Gamma \). Increasing \( \Gamma \) 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.

\section*{2.6 Robustness in Both Features and Labels}

In this section, we consider applying the methods of Sections 2.4 and 2.5 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 \textit{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 2.5, 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 \textit{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 Appendix.

Like both methods individually, the robust-in-both classifier has to select the robustness parameters $\rho$ and $\Gamma$ 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 $\rho/\Gamma$ 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.
2.6.1 Soft-Margin Support Vector Machines

Robustifying Problem (2.1) against both $\mathcal{U}_x$ and $\mathcal{U}_y$ gives the following robust optimization problem:

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

(2.27)

**Theorem 7.** The robust counterpart to Problem (2.27) is

$$
\min \sum_{i=1}^{n} \xi_i + \Gamma q + \sum_{i=1}^{n} r_i \\
\text{s.t.} \quad q + r_i \geq \phi_i - \xi_i \quad \forall i, \\
\xi_i \geq 1 - y_i(w^T x_i - b) + \rho \|w\|_{\ell^q}^* \quad \forall i, \\
\xi_i \leq 1 - y_i(w^T x_i - b) + \rho \|w\|_{\ell^q}^* + M(1 - s_i) \quad \forall i, \\
\xi_i \leq Ms_i \quad \forall i, \\
\phi_i \geq 1 + y_i(w^T x_i - b) + \rho \|w\|_{\ell^q}^* \quad \forall i, \\
\phi_i \leq 1 + y_i(w^T x_i - b) + \rho \|w\|_{\ell^q}^* + M(1 - t_i) \quad \forall i, \\
\phi_i \leq Mt_i \quad \forall i, \\
r_i, \xi_i, \phi_i \geq 0 \quad \forall i, \\
q \geq 0, \\
s, t \in \{0, 1\}^n.
$$

(2.28)

where $\ell_{q^*}$ is the dual norm of $\ell_q$, and $M$ is a sufficiently large constant.

The proof of Theorem 7 is straightforward, and it is provided in Appendix A.2.

Problem (2.28) is a mixed-integer optimization problem which is practically solvable.
2.6.2 Logistic Regression

Robustifying Problem (2.4) against both $\mathcal{U}_x$ and $\mathcal{U}_y$ gives the following robust optimization problem:

$$\max_{\beta, \beta_0} \min_{\Delta y \in \mathcal{U}_y} \min_{\Delta x \in \mathcal{U}_x} -\sum_{i=1}^{n} \log \left(1 + e^{-y_i(1 - 2\Delta y_i)(\beta^T(x_i + \Delta x_i) + \beta_0)}\right).$$  \hspace{1cm} (2.29)

**Theorem 8.** The robust counterpart to Problem (2.29) is

$$\max -\sum_{i=1}^{n} \log \left(1 + e^{-y_i(\beta^T x_i + \beta_0)} + \rho \|\beta\|_{q^*}\right) + \Gamma \mu + \sum_{i=1}^{n} \nu_i$$

s.t. $\mu + \nu_i \leq \log \left(\frac{1 + e^{-y_i(\beta^T x_i + \beta_0)} + \rho \|\beta\|_{q^*}}{1 + e^{y_i(\beta^T x_i + \beta_0)} + \rho \|\beta\|_{q^*}}\right)$ \hspace{1cm} \forall i,

$$\nu_i \leq 0 \hspace{1cm} \forall i,$$

$$\mu \leq 0,$$

where $\ell_{q^*}$ is the dual norm of $\ell_q$.

The proof of Theorem 8 can be found in Appendix A.2. It 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 (2.30) is a maximization of a concave, twice continuously differentiable function in $\beta$ and $\beta_0$ with constraints for any given $\rho$ and $\Gamma$. Therefore, we can solve this problem using interior point methods [9].

2.6.3 Optimal Decision Trees

Robustifying Problem (2.6) against both $\mathcal{U}_x$ and $\mathcal{U}_y$ gives a problem identical to Problem (2.6) with the following exceptions:

- The constraints in (2.16) in place of constraints (2.6p) and (2.6q);
- The constraints in (2.25) in place of constraints (2.6b), (2.6c), (2.6d), (2.6e), (2.6f).
Theorem 9. The robust counterpart to the above problem is identical to Problem (2.6) with the following exceptions:

- The constraints in (2.17) in place of constraints (2.6p) and (2.6q);
- The constraints in (2.26) in place of constraints (2.6b), (2.6c), (2.6d), (2.6e), (2.6f), and (2.6g).

The proof of Theorem 9 is given in Appendix A.2 and the complete robust-in-both formulation is stated in full.

This resulting problem is still a linear mixed-integer optimization problem, and so remains practically solvable.

2.7 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.

2.7.1 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, $N(1.5e, I)$, where $e$ is the vector of ones and $I$ is the identity matrix. These points are given the label $+1$. 


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

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

3. 10 outlier points are introduced as multivariate random normal, \( N(0, 3\textbf{I}) \), where \( \textbf{0} \) 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 2-2. 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 \( \textbf{e}^T \textbf{x} = 0 \), also shown in Figure 2-2. 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 ($e^T x = 0$).

### 2.7.2 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 [26]. The optimization problems required by each method were formulated in JuMP, a state-of-the-art library for algebraic modeling and mathematical optimization [87]. The commercial solver **GUROBI** [60] was used to solve the linear and mixed-integer optimization problems for SVM, and the open source solver **IPOPT** [127] was used to solve the convex optimization problems for logistic regression.

To ensure a fair comparison, we use the $\ell_1$ norm in the regularized methods and set $q = \infty$ for the feature uncertainty set so that the norms in the robust methods are also $\ell_1$ norms. For each method, the values of $\rho$ and $\Gamma$ were selected through validation when using the corresponding robust classifiers.

<table>
<thead>
<tr>
<th>Method</th>
<th>SVM</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Out-of-sample error (%)</td>
<td>Distance from truth</td>
</tr>
<tr>
<td>Nominal</td>
<td>2.571 ± 0.021</td>
<td>0.357 ± 0.004</td>
</tr>
<tr>
<td>Regularized</td>
<td>2.643 ± 0.027</td>
<td>0.357 ± 0.004</td>
</tr>
<tr>
<td>Features</td>
<td>2.516 ± 0.020</td>
<td>0.345 ± 0.004</td>
</tr>
<tr>
<td>Labels</td>
<td>2.396 ± 0.018</td>
<td>0.320 ± 0.004</td>
</tr>
<tr>
<td>Both</td>
<td>2.363 ± 0.018</td>
<td>0.310 ± 0.004</td>
</tr>
</tbody>
</table>

Table 2.1: 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.
### Table 2.2: Out-of-sample error results by percentile for synthetic data experiments.

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

### 2.7.3 Results

This experiment was repeated 2000 times, and we present the means and standard errors of the two measures for each method in Table 2.1. 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 2.2, we break down the results by percentile in out-of-sample error, and we report the 10\textsuperscript{th}, 20\textsuperscript{th}, \ldots, 90\textsuperscript{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 case scenario presented (90th 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.

2.8 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.

2.8.1 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 [79]. 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.

### 2.8.2 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 \( \ell_1 \). The implementations for SVM and logistic regression are identical to those used in the synthetic experiments, which are described in Section 2.7.1. We implement Optimal Decision Trees using the JuMP software package in Julia, and the commercial solver GUROBI \[60\] was used to solve the mixed-integer optimization problems.

As in the other two methods, for Optimal Decision Trees we select the values of \( \rho \) and \( \Gamma \) through validation when using the corresponding robust classifier. During validation, we also select the complexity parameter \( \text{cp} \), the minimum number of points per node \( \text{minbucket} \), and the exploration depth around the warm start solution \( \text{explorationdepth} \). See \[12\] for a full description of these parameters. We compare the robust counterparts of the Optimal
<table>
<thead>
<tr>
<th>Data Set Information</th>
<th>SVM</th>
<th>Logistic Regression</th>
<th>CART</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n</td>
<td>p</td>
<td>Nominal</td>
</tr>
<tr>
<td>acute-inflammations-1</td>
<td>120</td>
<td>7</td>
<td>1.0000</td>
</tr>
<tr>
<td>acute-inflammations-2</td>
<td>120</td>
<td>7</td>
<td>1.0000</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>68</td>
<td>280</td>
<td>0.5692</td>
</tr>
<tr>
<td>balance-scale</td>
<td>625</td>
<td>5</td>
<td>0.9200</td>
</tr>
<tr>
<td>banknote-authentication</td>
<td>1372</td>
<td>5</td>
<td>0.9912</td>
</tr>
<tr>
<td>blood-transfusion</td>
<td>748</td>
<td>5</td>
<td>0.7638</td>
</tr>
<tr>
<td>breast-cancer</td>
<td>683</td>
<td>10</td>
<td>0.9505</td>
</tr>
<tr>
<td>breast-cancer-diagnostic</td>
<td>569</td>
<td>31</td>
<td>0.9351</td>
</tr>
<tr>
<td>breast-cancer-prognostic</td>
<td>194</td>
<td>33</td>
<td>0.7128</td>
</tr>
<tr>
<td>car-evaluation</td>
<td>1728</td>
<td>16</td>
<td>0.8000</td>
</tr>
<tr>
<td>chess-king-rook-vs-king</td>
<td>28056</td>
<td>35</td>
<td>0.9004</td>
</tr>
<tr>
<td>climate-model-crashes</td>
<td>540</td>
<td>19</td>
<td>0.9500</td>
</tr>
<tr>
<td>cnae-9</td>
<td>1080</td>
<td>857</td>
<td>0.9750</td>
</tr>
<tr>
<td>congressional-voting-records</td>
<td>232</td>
<td>17</td>
<td>0.9565</td>
</tr>
<tr>
<td>connectionist-bench</td>
<td>990</td>
<td>11</td>
<td>0.9758</td>
</tr>
<tr>
<td>connectionist-bench-sonar</td>
<td>208</td>
<td>61</td>
<td>0.7073</td>
</tr>
<tr>
<td>contraceptive-method-choice</td>
<td>1473</td>
<td>12</td>
<td>0.6776</td>
</tr>
<tr>
<td>credit-approval</td>
<td>653</td>
<td>38</td>
<td>0.8508</td>
</tr>
<tr>
<td>cylinder-bands</td>
<td>277</td>
<td>485</td>
<td>0.5564</td>
</tr>
<tr>
<td>dermatology</td>
<td>358</td>
<td>35</td>
<td>0.9662</td>
</tr>
<tr>
<td>echocardiogram</td>
<td>61</td>
<td>7</td>
<td>0.7167</td>
</tr>
<tr>
<td>ecoli</td>
<td>336</td>
<td>8</td>
<td>0.9582</td>
</tr>
<tr>
<td>fertility</td>
<td>100</td>
<td>13</td>
<td>0.8700</td>
</tr>
<tr>
<td>flags</td>
<td>194</td>
<td>60</td>
<td>0.6923</td>
</tr>
</tbody>
</table>

Table 2.3: Out-of-sample accuracy averaged across five seeds for each classification method and its robust counterparts on all data sets. For each data set, the best result is indicated in **bold**, and the best method overall for the data set is **underlined**.
<table>
<thead>
<tr>
<th>UCI Data Set Name</th>
<th>n</th>
<th>p</th>
<th>SVM Nominal Features</th>
<th>SVM Labels</th>
<th>Logistic Regression Nominal Features</th>
<th>Logistic Regression Labels</th>
<th>CART Nominal Features</th>
<th>CART Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>glass-identification</td>
<td>214</td>
<td>10</td>
<td>0.7163</td>
<td>0.7395</td>
<td>0.7116</td>
<td>0.7488</td>
<td>0.7746</td>
<td>0.7860</td>
</tr>
<tr>
<td>haberman-survival</td>
<td>306</td>
<td>4</td>
<td>0.7279</td>
<td>0.7344</td>
<td>0.7344</td>
<td>0.7410</td>
<td>0.7049</td>
<td>0.6623</td>
</tr>
<tr>
<td>hayes-roth</td>
<td>132</td>
<td>5</td>
<td>0.6846</td>
<td>0.6846</td>
<td>0.6615</td>
<td>0.8000</td>
<td>0.8154</td>
<td>0.8154</td>
</tr>
<tr>
<td>heart-disease-cleveland</td>
<td>297</td>
<td>19</td>
<td>0.8407</td>
<td>0.8339</td>
<td>0.8305</td>
<td>0.8339</td>
<td>0.7759</td>
<td>0.7095</td>
</tr>
<tr>
<td>hepatitis</td>
<td>80</td>
<td>20</td>
<td>0.8500</td>
<td>0.8500</td>
<td>0.8375</td>
<td>0.8250</td>
<td>0.8125</td>
<td>0.7875</td>
</tr>
<tr>
<td>hill-valley</td>
<td>606</td>
<td>101</td>
<td>0.5884</td>
<td>0.9620</td>
<td>0.9934</td>
<td>0.9636</td>
<td>0.5504</td>
<td>0.5504</td>
</tr>
<tr>
<td>hill-valley-noise</td>
<td>606</td>
<td>101</td>
<td>0.8612</td>
<td>0.8545</td>
<td>0.8463</td>
<td>0.8876</td>
<td>0.4744</td>
<td>0.4992</td>
</tr>
<tr>
<td>image-segmentation</td>
<td>210</td>
<td>20</td>
<td>0.9286</td>
<td>0.9857</td>
<td>0.9762</td>
<td>0.9762</td>
<td>0.9476</td>
<td>0.9810</td>
</tr>
<tr>
<td>indian-liver-patient</td>
<td>579</td>
<td>11</td>
<td>0.7155</td>
<td>0.7155</td>
<td>0.7172</td>
<td>0.7224</td>
<td>0.6931</td>
<td>0.6862</td>
</tr>
<tr>
<td>ionosphere</td>
<td>351</td>
<td>35</td>
<td>0.8743</td>
<td>0.8743</td>
<td>0.8829</td>
<td>0.8743</td>
<td>0.8971</td>
<td>0.9086</td>
</tr>
<tr>
<td>iris</td>
<td>150</td>
<td>5</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>letter-recognition</td>
<td>20000</td>
<td>17</td>
<td>0.9921</td>
<td>0.9922</td>
<td>0.9923</td>
<td>0.9923</td>
<td>0.9908</td>
<td>0.9896</td>
</tr>
<tr>
<td>libras-movement</td>
<td>360</td>
<td>91</td>
<td>0.9056</td>
<td>0.9694</td>
<td>0.9444</td>
<td>0.9639</td>
<td>0.9333</td>
<td>0.9528</td>
</tr>
<tr>
<td>magic-gamma-telecope</td>
<td>19020</td>
<td>11</td>
<td>0.7922</td>
<td>0.7923</td>
<td>0.7924</td>
<td>0.7924</td>
<td>0.8364</td>
<td>0.8367</td>
</tr>
<tr>
<td>mammographic-mass</td>
<td>830</td>
<td>11</td>
<td>0.8193</td>
<td>0.8072</td>
<td>0.8289</td>
<td>0.8289</td>
<td>0.8145</td>
<td>0.8108</td>
</tr>
<tr>
<td>monks-problems-1</td>
<td>124</td>
<td>12</td>
<td>0.8240</td>
<td>0.7360</td>
<td>0.7440</td>
<td>0.7680</td>
<td>0.8080</td>
<td>0.8400</td>
</tr>
<tr>
<td>monks-problems-2</td>
<td>169</td>
<td>12</td>
<td>0.6118</td>
<td>0.6176</td>
<td>0.5647</td>
<td>0.6235</td>
<td>0.6118</td>
<td>0.6294</td>
</tr>
<tr>
<td>monks-problems-3</td>
<td>122</td>
<td>12</td>
<td>0.9167</td>
<td>0.9333</td>
<td>0.8500</td>
<td>0.9250</td>
<td>0.8917</td>
<td>0.9333</td>
</tr>
<tr>
<td>mushroom</td>
<td>5644</td>
<td>77</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9894</td>
<td>0.9901</td>
</tr>
<tr>
<td>nursery</td>
<td>12960</td>
<td>20</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9894</td>
<td>0.9901</td>
</tr>
<tr>
<td>optical-recognition</td>
<td>3823</td>
<td>65</td>
<td>0.9929</td>
<td>0.9961</td>
<td>0.9929</td>
<td>0.9976</td>
<td>0.9830</td>
<td>0.9830</td>
</tr>
<tr>
<td>ozone-level-detection-eight</td>
<td>1847</td>
<td>73</td>
<td>0.9301</td>
<td>0.9301</td>
<td>0.9312</td>
<td>0.9322</td>
<td>0.9322</td>
<td>0.9328</td>
</tr>
<tr>
<td>ozone-level-detection-one</td>
<td>1848</td>
<td>73</td>
<td>0.9545</td>
<td>0.9702</td>
<td>0.9561</td>
<td>0.9702</td>
<td>0.9702</td>
<td>0.9702</td>
</tr>
<tr>
<td>parkinsons</td>
<td>195</td>
<td>22</td>
<td>0.8564</td>
<td>0.8359</td>
<td>0.8410</td>
<td>0.8410</td>
<td>0.8410</td>
<td>0.8410</td>
</tr>
</tbody>
</table>

Table 2.3: (cont.) Out-of-sample accuracy averaged across five seeds for each classification method and its robust counterparts on all data sets. For each data set, the best result is indicated in **bold**, and the best method overall for the data set is *underlined.*
Table 2.3: (cont.) Out-of-sample accuracy averaged across five seeds for each classification method and its robust counterparts on all data sets. For each data set, the best result is indicated in **bold**, and the best method overall for the data set is underlined.
| UCI Data Set Name | n  | p  | Nominal Features | Labels | Both | Nominal Features | Labels | Both | Nominal Features | Labels | Both | Nominal Features | Labels | Both | Nominal Features | Labels | Both |
|-------------------|----|----|------------------|--------|------|------------------|--------|------|------------------|--------|------|------------------|--------|------|------------------|--------|------|------------------|--------|------|
| yeast             | 1484 | 9  | 0.6902           | 0.6902 | 0.6902 | 0.6801           | 0.6828 | 0.6929 | 0.6929           | 0.7286 | 0.7219 | 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 |                           |

Table 2.3: (cont.) Out-of-sample accuracy averaged across five seeds for each classification method and its robust counterparts on all data sets. For each data set, the best result is indicated in **bold**, and the best method overall for the data set is underlined.
Figure 2-3: Pairwise comparisons between nominal and individual robust methods. For each type of robustness, the plots compare that particular robust method and the nominal method and show the number of data sets for which each had the highest out-of-sample accuracy.

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 [118] in the R programming language [104].

Table 2.3 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.

### 2.8.3 Pairwise Comparisons

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

Results for the three nominal methods and all robust variations are summarized in Figure 2-3. Each pair of bars in the graph represents 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.
<table>
<thead>
<tr>
<th>Nominal Method</th>
<th>Robustness Type</th>
<th>Wins</th>
<th>Losses</th>
<th>Ties</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>Features</td>
<td>37</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Labels</td>
<td>35</td>
<td>18</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Both</td>
<td>39</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>Features</td>
<td>34</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Labels</td>
<td>35</td>
<td>21</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Both</td>
<td>40</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>CART</td>
<td>Features</td>
<td>36</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Labels</td>
<td>33</td>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Both</td>
<td>33</td>
<td>24</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 2.4: Pairwise comparisons of robust classification methods against their nominal counterparts.

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 2.4.

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 2-4 shows the breakdown of counts for data sets in which there is a unique highest out-of-sample accuracy. All 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.

### 2.8.4 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 2.5 shows the improvement in out-of-sample accuracy of robust-in-both methods 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
Table 2.5: Improvement due to robustness by baseline in-sample accuracy, comparing the baseline method to the corresponding robust-in-both classifier.

<table>
<thead>
<tr>
<th>Nominal Method</th>
<th>Nominal Accuracy</th>
<th>Wins</th>
<th>Losses</th>
<th>Ties</th>
<th>Robust Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SVM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0–60%</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td></td>
<td>10.7 ± 5.6%</td>
</tr>
<tr>
<td>60–70%</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td></td>
<td>2.2 ± 1.9%</td>
</tr>
<tr>
<td>70–80%</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td></td>
<td>0.9 ± 1.1%</td>
</tr>
<tr>
<td>80–90%</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td></td>
<td>0.3 ± 0.5%</td>
</tr>
<tr>
<td>90–100%</td>
<td>14</td>
<td>9</td>
<td>11</td>
<td></td>
<td>0.0 ± 0.4%</td>
</tr>
<tr>
<td><strong>Logistic Regression</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0–60%</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td>7.7 ± 5.2%</td>
</tr>
<tr>
<td>60–70%</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td></td>
<td>4.5 ± 1.2%</td>
</tr>
<tr>
<td>70–80%</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td></td>
<td>1.2 ± 1.0%</td>
</tr>
<tr>
<td>80–90%</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td></td>
<td>1.1 ± 0.9%</td>
</tr>
<tr>
<td>90–100%</td>
<td>16</td>
<td>7</td>
<td>14</td>
<td></td>
<td>0.2 ± 0.1%</td>
</tr>
<tr>
<td><strong>CART</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0–60%</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td></td>
<td>0.7 ± 0.7%</td>
</tr>
<tr>
<td>60–70%</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>2.4 ± −%</td>
</tr>
<tr>
<td>70–80%</td>
<td>7</td>
<td>8</td>
<td>2</td>
<td></td>
<td>0.1 ± 0.9%</td>
</tr>
<tr>
<td>80–90%</td>
<td>7</td>
<td>8</td>
<td>0</td>
<td></td>
<td>−0.3 ± 0.9%</td>
</tr>
<tr>
<td>90–100%</td>
<td>17</td>
<td>8</td>
<td>14</td>
<td></td>
<td>−0.1 ± 0.6%</td>
</tr>
</tbody>
</table>

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.

Next, we consider the combined influence of nominal in-sample accuracy and dimension of data points on the robust improvement. Figure 2-5 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 2.6 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
Figure 2-5: 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 • and ×, respectively.
Table 2.6: Improvement due to robustness by baseline in-sample accuracy and dimension of points, comparing the nominal method to the corresponding robust-in-both classifier. 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).

Table 2.6 shows that robust methods perform strongest relative to nominal methods, even in the presence of a significantly stronger nominal method.

We also include in Table 2.6 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 formulation 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:

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

(2.31)

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 2.6 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, (2.31) gives a simple but strong recommendation for when to use robust classification in place of nominal SVM, logistic regression, or CART.

### 2.8.5 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>
<thead>
<tr>
<th>UCI Data Set Name</th>
<th>n</th>
<th>p</th>
<th>SVM</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Regularized</td>
<td>Robust</td>
</tr>
<tr>
<td>acute-inflammations-1</td>
<td>120</td>
<td>7</td>
<td>1.0000</td>
<td>0.9083</td>
</tr>
<tr>
<td>acute-inflammations-2</td>
<td>120</td>
<td>7</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>68</td>
<td>280</td>
<td>0.6154</td>
<td>0.6308</td>
</tr>
<tr>
<td>balance-scale</td>
<td>625</td>
<td>5</td>
<td>0.9200</td>
<td>0.9200</td>
</tr>
<tr>
<td>banknote-authentication</td>
<td>1372</td>
<td>5</td>
<td>0.9869</td>
<td>0.9912</td>
</tr>
<tr>
<td>blood-transfusion</td>
<td>748</td>
<td>5</td>
<td>0.7638</td>
<td>0.7638</td>
</tr>
<tr>
<td>breast-cancer</td>
<td>683</td>
<td>10</td>
<td>0.9679</td>
<td>0.9559</td>
</tr>
<tr>
<td>breast-cancer-diagnostic</td>
<td>569</td>
<td>31</td>
<td>0.9719</td>
<td>0.9614</td>
</tr>
<tr>
<td>breast-cancer-prognostic</td>
<td>194</td>
<td>33</td>
<td>0.7692</td>
<td>0.7179</td>
</tr>
<tr>
<td>car-evaluation</td>
<td>1728</td>
<td>16</td>
<td>0.7977</td>
<td>0.7826</td>
</tr>
<tr>
<td>chess-king-rook-vs-king</td>
<td>28056</td>
<td>35</td>
<td>0.9004</td>
<td>0.9004</td>
</tr>
<tr>
<td>chess-king-rook-vs-king-pawn</td>
<td>3196</td>
<td>38</td>
<td>0.9743</td>
<td>0.9687</td>
</tr>
<tr>
<td>climate-model-crashes</td>
<td>540</td>
<td>19</td>
<td>0.9611</td>
<td>0.9574</td>
</tr>
<tr>
<td>cnae-9</td>
<td>1080</td>
<td>857</td>
<td>0.9769</td>
<td>0.9481</td>
</tr>
<tr>
<td>congressional-voting-records</td>
<td>232</td>
<td>17</td>
<td>0.9787</td>
<td>0.9826</td>
</tr>
<tr>
<td>connectionist-bench</td>
<td>990</td>
<td>11</td>
<td>0.9737</td>
<td>0.9768</td>
</tr>
<tr>
<td>connectionist-bench-sonar</td>
<td>298</td>
<td>61</td>
<td>0.7268</td>
<td>0.7561</td>
</tr>
<tr>
<td>contraceptive-method-choice</td>
<td>1473</td>
<td>12</td>
<td>0.6800</td>
<td>0.6755</td>
</tr>
<tr>
<td>credit-approval</td>
<td>653</td>
<td>38</td>
<td>0.8626</td>
<td>0.8585</td>
</tr>
<tr>
<td>cylinder-bands</td>
<td>277</td>
<td>485</td>
<td>0.7200</td>
<td>0.6691</td>
</tr>
<tr>
<td>dermatology</td>
<td>358</td>
<td>35</td>
<td>0.9915</td>
<td>0.9803</td>
</tr>
<tr>
<td>echocardiogram</td>
<td>61</td>
<td>7</td>
<td>0.7000</td>
<td>0.6833</td>
</tr>
<tr>
<td>ecoli</td>
<td>336</td>
<td>8</td>
<td>0.9791</td>
<td>0.9582</td>
</tr>
<tr>
<td>fertility</td>
<td>100</td>
<td>13</td>
<td>0.8500</td>
<td>0.9000</td>
</tr>
<tr>
<td>flags</td>
<td>194</td>
<td>60</td>
<td>0.8872</td>
<td>0.8205</td>
</tr>
<tr>
<td>glass-identification</td>
<td>214</td>
<td>10</td>
<td>0.7302</td>
<td>0.7256</td>
</tr>
<tr>
<td>Haberman-survival</td>
<td>306</td>
<td>4</td>
<td>0.7279</td>
<td>0.7344</td>
</tr>
<tr>
<td>Hayes-roth</td>
<td>132</td>
<td>5</td>
<td>0.8519</td>
<td>0.6692</td>
</tr>
<tr>
<td>heart-disease-cleveland</td>
<td>297</td>
<td>19</td>
<td>0.8305</td>
<td>0.8203</td>
</tr>
<tr>
<td>hepatitis</td>
<td>80</td>
<td>20</td>
<td>0.8375</td>
<td>0.8125</td>
</tr>
<tr>
<td>hill-valley</td>
<td>606</td>
<td>101</td>
<td>0.8364</td>
<td>0.9620</td>
</tr>
<tr>
<td>hill-valley-noise</td>
<td>606</td>
<td>101</td>
<td>0.8132</td>
<td>0.8512</td>
</tr>
<tr>
<td>image-segmentation</td>
<td>210</td>
<td>20</td>
<td>0.9905</td>
<td>0.9476</td>
</tr>
<tr>
<td>Indian-liver-patient</td>
<td>579</td>
<td>11</td>
<td>0.7155</td>
<td>0.7155</td>
</tr>
<tr>
<td>ionosphere</td>
<td>351</td>
<td>35</td>
<td>0.8743</td>
<td>0.8743</td>
</tr>
<tr>
<td>iris</td>
<td>150</td>
<td>5</td>
<td>1.0000</td>
<td>0.9800</td>
</tr>
<tr>
<td>letter-recognition</td>
<td>20000</td>
<td>17</td>
<td>0.9916</td>
<td>0.9923</td>
</tr>
</tbody>
</table>

Table 2.7: Out-of-sample accuracy averaged across five seeds for each method using both regularized and robust-in-both methods on all data sets. For each data set, the best result is indicated in **bold**.
<table>
<thead>
<tr>
<th>UCI Data Set Name</th>
<th>$n$</th>
<th>$p$</th>
<th>SVM Regularized</th>
<th>SVM Robust</th>
<th>Logistic Regression Regularized</th>
<th>Logistic Regression Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td>libras-movement</td>
<td>360</td>
<td>91</td>
<td><strong>0.9694</strong></td>
<td><strong>0.9694</strong></td>
<td>0.9583</td>
<td>0.9639</td>
</tr>
<tr>
<td>magic-gamma-telescope</td>
<td>19020</td>
<td>11</td>
<td>0.7848</td>
<td><strong>0.7924</strong></td>
<td>0.7862</td>
<td><strong>0.7919</strong></td>
</tr>
<tr>
<td>mammographic-mass</td>
<td>830</td>
<td>11</td>
<td><strong>0.8120</strong></td>
<td>0.8060</td>
<td><strong>0.8301</strong></td>
<td>0.8217</td>
</tr>
<tr>
<td>monks-problems-1</td>
<td>124</td>
<td>12</td>
<td>0.6960</td>
<td><strong>0.8000</strong></td>
<td>0.6560</td>
<td><strong>0.7920</strong></td>
</tr>
<tr>
<td>monks-problems-2</td>
<td>169</td>
<td>12</td>
<td>0.5824</td>
<td><strong>0.6176</strong></td>
<td>0.5882</td>
<td><strong>0.6235</strong></td>
</tr>
<tr>
<td>monks-problems-3</td>
<td>122</td>
<td>12</td>
<td><strong>0.9360</strong></td>
<td>0.9333</td>
<td><strong>0.9360</strong></td>
<td>0.9250</td>
</tr>
<tr>
<td>mushroom</td>
<td>5644</td>
<td>77</td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
</tr>
<tr>
<td>nursery</td>
<td>12960</td>
<td>20</td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
</tr>
<tr>
<td>optical-recognition</td>
<td>3823</td>
<td>65</td>
<td>0.9956</td>
<td><strong>0.9966</strong></td>
<td>0.9958</td>
<td><strong>0.9971</strong></td>
</tr>
<tr>
<td>ozone-level-detection-eight</td>
<td>1847</td>
<td>73</td>
<td><strong>0.9355</strong></td>
<td>0.9295</td>
<td><strong>0.9366</strong></td>
<td>0.9317</td>
</tr>
<tr>
<td>ozone-level-detection-one</td>
<td>1848</td>
<td>73</td>
<td><strong>0.9702</strong></td>
<td><strong>0.9702</strong></td>
<td>0.9675</td>
<td><strong>0.9702</strong></td>
</tr>
<tr>
<td>parkinsons</td>
<td>195</td>
<td>22</td>
<td><strong>0.8872</strong></td>
<td>0.8615</td>
<td><strong>0.8462</strong></td>
<td>0.8205</td>
</tr>
<tr>
<td>pen-based-recognition</td>
<td>7494</td>
<td>17</td>
<td><strong>0.9893</strong></td>
<td>0.9891</td>
<td>0.9896</td>
<td><strong>0.9897</strong></td>
</tr>
<tr>
<td>pima-indians-diabetes</td>
<td>768</td>
<td>9</td>
<td>0.7647</td>
<td><strong>0.7791</strong></td>
<td>0.7660</td>
<td><strong>0.7752</strong></td>
</tr>
<tr>
<td>planning-relax</td>
<td>182</td>
<td>13</td>
<td>0.7027</td>
<td><strong>0.7222</strong></td>
<td><strong>0.7027</strong></td>
<td>0.7000</td>
</tr>
<tr>
<td>poker-hand</td>
<td>25010</td>
<td>11</td>
<td>0.5018</td>
<td><strong>0.5023</strong></td>
<td>0.5005</td>
<td>0.5000</td>
</tr>
<tr>
<td>post-operative-patient</td>
<td>87</td>
<td>14</td>
<td><strong>0.7059</strong></td>
<td><strong>0.7059</strong></td>
<td><strong>0.7059</strong></td>
<td>0.6588</td>
</tr>
<tr>
<td>qsar-biodegradation</td>
<td>1055</td>
<td>42</td>
<td>0.8692</td>
<td><strong>0.8758</strong></td>
<td>0.8578</td>
<td><strong>0.8682</strong></td>
</tr>
<tr>
<td>seeds</td>
<td>210</td>
<td>8</td>
<td>0.9333</td>
<td><strong>0.9429</strong></td>
<td><strong>0.9619</strong></td>
<td>0.9476</td>
</tr>
<tr>
<td>seismic-bumps</td>
<td>2584</td>
<td>21</td>
<td><strong>0.9342</strong></td>
<td><strong>0.9342</strong></td>
<td><strong>0.9335</strong></td>
<td>0.9327</td>
</tr>
<tr>
<td>skin-segmentation</td>
<td>24507</td>
<td>4</td>
<td>0.9326</td>
<td><strong>0.9366</strong></td>
<td>0.9187</td>
<td><strong>0.9345</strong></td>
</tr>
<tr>
<td>soybean-large</td>
<td>266</td>
<td>63</td>
<td><strong>0.9094</strong></td>
<td>0.8717</td>
<td><strong>0.9170</strong></td>
<td>0.8868</td>
</tr>
<tr>
<td>soybean-small</td>
<td>47</td>
<td>38</td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
<td><strong>1.0000</strong></td>
</tr>
<tr>
<td>spambase</td>
<td>4601</td>
<td>58</td>
<td>0.9287</td>
<td>0.9265</td>
<td>0.9241</td>
<td><strong>0.9246</strong></td>
</tr>
<tr>
<td>spect-heart</td>
<td>80</td>
<td>23</td>
<td>0.6375</td>
<td><strong>0.7125</strong></td>
<td>0.6750</td>
<td><strong>0.7625</strong></td>
</tr>
<tr>
<td>spectf-heart</td>
<td>80</td>
<td>45</td>
<td>0.6375</td>
<td><strong>0.6500</strong></td>
<td>0.6750</td>
<td><strong>0.7625</strong></td>
</tr>
<tr>
<td>statlog-project-german-credit</td>
<td>1000</td>
<td>49</td>
<td><strong>0.7420</strong></td>
<td>0.7400</td>
<td>0.7350</td>
<td><strong>0.7400</strong></td>
</tr>
<tr>
<td>statlog-project-landsat-satellite</td>
<td>4435</td>
<td>37</td>
<td><strong>0.9867</strong></td>
<td>0.9820</td>
<td><strong>0.9851</strong></td>
<td>0.9824</td>
</tr>
<tr>
<td>teaching-assistant-evaluation</td>
<td>151</td>
<td>53</td>
<td><strong>0.7200</strong></td>
<td>0.6733</td>
<td><strong>0.8067</strong></td>
<td>0.7067</td>
</tr>
<tr>
<td>thoracic-surgery</td>
<td>470</td>
<td>25</td>
<td><strong>0.8511</strong></td>
<td><strong>0.8511</strong></td>
<td><strong>0.8532</strong></td>
<td>0.8468</td>
</tr>
<tr>
<td>thyroid-disease-ann-thyroid</td>
<td>3772</td>
<td>22</td>
<td>0.9905</td>
<td><strong>0.9915</strong></td>
<td>0.9910</td>
<td><strong>0.9934</strong></td>
</tr>
<tr>
<td>thyroid-disease-new-thyroid</td>
<td>215</td>
<td>6</td>
<td>0.8837</td>
<td><strong>0.8884</strong></td>
<td>0.8977</td>
<td><strong>0.8977</strong></td>
</tr>
<tr>
<td>tic-tac-toe-endgame</td>
<td>958</td>
<td>19</td>
<td><strong>0.9812</strong></td>
<td>0.9801</td>
<td><strong>0.9801</strong></td>
<td>0.9801</td>
</tr>
<tr>
<td>wall-following-robot-2</td>
<td>5456</td>
<td>3</td>
<td><strong>0.6440</strong></td>
<td>0.5553</td>
<td>0.6537</td>
<td><strong>0.6609</strong></td>
</tr>
<tr>
<td>wall-following-robot-24</td>
<td>5456</td>
<td>5</td>
<td>0.6436</td>
<td><strong>0.6561</strong></td>
<td><strong>0.6565</strong></td>
<td>0.6563</td>
</tr>
<tr>
<td>wine</td>
<td>178</td>
<td>14</td>
<td><strong>0.9829</strong></td>
<td>0.9657</td>
<td>0.9886</td>
<td><strong>0.9943</strong></td>
</tr>
<tr>
<td>yeast</td>
<td>1484</td>
<td>9</td>
<td>0.6869</td>
<td><strong>0.6902</strong></td>
<td>0.6842</td>
<td><strong>0.6929</strong></td>
</tr>
</tbody>
</table>

Table 2.7: (cont.) Out-of-sample accuracy averaged across five seeds for each method using both regularized and robust-in-both methods on all data sets. For each data set, the best result is indicated in **bold**.
First, we compare our feature-robust SVM to $\ell_1$-regularized SVM, which is equivalent to classical SVM except for the $\ell_1$ norm regularizer term. This is a feature-robust method under a different uncertainty set (see Section 2.4.2). We implemented Problem (2.3) in JuMP and solved this problem with Gurobi. Experimentally, feature-robust SVM and $\ell_1$-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 $\ell_1$-regularized SVM, and robust-in-both logistic regression against $\ell_1$-regularized logistic regression (which uses an ad-hoc method for introducing robustness). For $\ell_1$-regularized logistic regression, we implemented Problem (2.5) with $q = 1$ in JuMP and solved this problem with IPOPT. We present the accuracy results for this comparison in Table 2.7.

In Table 2.7, we present the relative performance of robust-in-both and regularized methods broken down into the same two regions as defined in Section 2.8.4. 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 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
<table>
<thead>
<tr>
<th>Baseline Method</th>
<th>Region</th>
<th>Wins</th>
<th>Losses</th>
<th>Ties</th>
<th>Robust Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regularized SVM</td>
<td>Above</td>
<td>8</td>
<td>6</td>
<td>1</td>
<td>0.5 ± 1.1%</td>
</tr>
<tr>
<td></td>
<td>Below</td>
<td>18</td>
<td>29</td>
<td>13</td>
<td>−0.7 ± 0.5%</td>
</tr>
<tr>
<td>Regularized Logistic Regression</td>
<td>Above</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>1.9 ± 1.6%</td>
</tr>
<tr>
<td></td>
<td>Below</td>
<td>24</td>
<td>28</td>
<td>10</td>
<td>0.1 ± 0.3%</td>
</tr>
</tbody>
</table>

Table 2.8: Improvement due to robustness by baseline in-sample accuracy and dimension of points, comparing the regularized method to the corresponding robust-in-both classifier. Region Above refers to the top-left sections in Figure 2-5 (high data dimension, low baseline accuracy); Region Below refers to the bottom-right sections in Figure 2-5 (low data dimension, high baseline accuracy).

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 (2.31).

### 2.8.6 Computational Tractability and Speed

Table 2.9 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. 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
Table 2.9: Problem complexity of nominal and robust classification methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Nominal</th>
<th>Feature-robust</th>
<th>Label-robust</th>
<th>Robust-in-both</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>LO</td>
<td>LO</td>
<td>MIO</td>
<td>MIO</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>Unconstr. CO</td>
<td>Unconstr. CO</td>
<td>Constr. CO</td>
<td>Constr. CO</td>
</tr>
<tr>
<td>Decision Trees</td>
<td>MIO</td>
<td>MIO</td>
<td>MIO</td>
<td>MIO</td>
</tr>
</tbody>
</table>

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 2.10. 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.

2.8.7 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 [23]. 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...
Table 2.10: Solver time for selected UCI data sets in seconds given robustness parameters $\rho = 0.01$ and $\Gamma = 10\%$. 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.
2.9 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.

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.
Chapter 3

Optimal Missing Data Imputation

This work appeared in the Journal of Machine Learning Research, with co-authors Dimitris Bertsimas and Ying Daisy Zhuo [22].

Missing data is a common problem in real-world settings and for this reason has attracted significant attention in the statistical literature. In this chapter, 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 in this chapter. We demonstrate that our proposed method improves out-of-sample accuracy in large-scale computational experiments across a sample of 84 data sets taken from the UCI Machine Learning Repository. In all scenarios of missing at random mechanisms and various missing percentages, opt.impute produces the best overall imputation in most data sets benchmarked against five other methods: mean impute, $K$-nearest neighbors, iterative knn, Bayesian PCA, and predictive-mean matching, with an average reduction in mean absolute error of 8.3% against the best cross-validated benchmark 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 0.339 in the regression tasks and the average out-of-sample accuracy is 86.1% in the classification tasks, compared to 0.315 and 84.4% for the best cross-validated benchmark method. In the multiple imputation setting, downstream models trained using opt.impute obtain a statistically significant improvement over models trained using multivariate imputation by chained equations (mice) in 8/10 missing data scenarios considered.

3.1 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 [82]. 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 [32, 128].

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 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. In this work, we focus on methods for missing data imputation.

Concretely, assume that we are given data $X = \{x_1, \ldots, x_n\}$ 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
Table 3.1: List of Imputation Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Category</th>
<th>Software</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean impute (mean)</td>
<td>Mean</td>
<td></td>
<td>[82]</td>
</tr>
<tr>
<td>Expectation-Maximization (EM)</td>
<td>EM</td>
<td></td>
<td>[49]</td>
</tr>
<tr>
<td>EM with Mixture of Gaussians and Multinomials</td>
<td>EM</td>
<td></td>
<td>[58]</td>
</tr>
<tr>
<td>EM with Bootstrapping</td>
<td>EM</td>
<td>Amelia II</td>
<td>[63]</td>
</tr>
<tr>
<td>K-Nearest Neighbors (knn)</td>
<td>K-NN</td>
<td>impute</td>
<td>[122]</td>
</tr>
<tr>
<td>Sequential K-Nearest Neighbors</td>
<td>K-NN</td>
<td></td>
<td>[71]</td>
</tr>
<tr>
<td>Iterative K-Nearest Neighbors</td>
<td>K-NN</td>
<td></td>
<td>[11, 51]</td>
</tr>
<tr>
<td>Support Vector Regression</td>
<td>SVR</td>
<td></td>
<td>[129]</td>
</tr>
<tr>
<td>Predictive-Mean Matching (pmm)</td>
<td>LS</td>
<td>MICE</td>
<td>[122]</td>
</tr>
<tr>
<td>Least Squares</td>
<td>LS</td>
<td></td>
<td>[30]</td>
</tr>
<tr>
<td>Sequential Regression Multivariate Imputation</td>
<td>LS</td>
<td></td>
<td>[103]</td>
</tr>
<tr>
<td>Local-Least Squares</td>
<td>LS</td>
<td></td>
<td>[70]</td>
</tr>
<tr>
<td>Sequential Local-Least Squares</td>
<td>LS</td>
<td></td>
<td>[130]</td>
</tr>
<tr>
<td>Iterative Local-Least Squares</td>
<td>LS</td>
<td></td>
<td>[33]</td>
</tr>
<tr>
<td>Sequential Regression Trees</td>
<td>Tree</td>
<td>MICE</td>
<td>[33]</td>
</tr>
<tr>
<td>Sequential Random Forest</td>
<td>Tree</td>
<td>missForest</td>
<td>[113]</td>
</tr>
<tr>
<td>Singular Value Decomposition</td>
<td>SVD</td>
<td></td>
<td>[122]</td>
</tr>
<tr>
<td>Bayesian Principal Component Analysis</td>
<td>SVD</td>
<td>pcaMethods</td>
<td>[99, 72]</td>
</tr>
<tr>
<td>Factor Analysis Model for Mixed Data</td>
<td>FA</td>
<td></td>
<td>[60]</td>
</tr>
</tbody>
</table>

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 3.1 and describe them briefly below. Part of the list is adapted from a review paper by [80].

3.1.1 Related Work

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 [82].

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 [63]. Joint modeling provides useful theoretical properties but lacks the flexibility for processing data types seen in many real applications [124]. 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 \cite{58}.

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 \cite{124}. To generalize to multivariate settings, a chained equation process — initializing using random sampling and conducting univariate imputations sequentially until convergence — is typically used \cite{125}. 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 \cite{30,105}. Alternatively, in the Predictive-Mean Matching method (pmm), imputations are random samples drawn from a set of observed values close to regression predictions \cite{125}. Imputation methods that use Support Vector Regression in place of LS for the regression step have also been explored \cite{129}.

When there is non-linear relationship between the variables, linear regression based imputation may perform poorly. \cite{33} propose using Classification and Regression Trees (CART) as the conditional model for imputation. Extensions to random forests have also shown promising results \cite{113}. 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 $d$th dimension of the $K$-nearest neighbors that have observed values in dimension $d$ \cite{122}. 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 \cite{71}. Iterative $K$-NN uses an iterative process to refine the estimates and choose the nearest neighbors based on the estimates from the previous iteration \cite{41,31}. 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 $x_i$ [70]. Sequential and iterative variations of Local-Least Squares resemble their $K$-NN imputation counterparts [139, 35].

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 [122, 89]. Bayesian Principal Component Analysis is similar to SVD imputation but extends the method to incorporate information from a prior distribution on the model parameters [99, 92]. Some recent development of a variant of the EM algorithm for factor analysis also provides a missing data imputation method for mixed data [69].

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 [82]. 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 [123]. 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 \cite{63}. Some conditional specification methods such as \texttt{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 \cite{25,19}. Recent work on Optimal Decision Trees for classification leverages integer and robust optimization \cite{12,14}. 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.

### 3.1.2 Contributions

We summarize our contributions in this chapter 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 functions 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 \texttt{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 84 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, \texttt{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 \texttt{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 \texttt{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 \texttt{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 \texttt{opt.impute} significantly outperform multiple imputations produced by \texttt{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 \ref{sec:methods}, we formulate the missing data imputation problem as an optimization problem, present a general first-order method \texttt{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 \texttt{opt.impute} to multiple imputation. In Section \ref{sec:results}, we compare the imputation quality and performance on downstream tasks of \texttt{opt.impute} to benchmark imputation methods on a wide range of real data sets. In Section \ref{sec:discussion}, we discuss the benefits from adopting such framework and suggest areas for future work. We conclude in Section \ref{sec:conclusion}.

### 3.2 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 data set 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 $d$th 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:

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

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.

### 3.2.1 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 $z_{ij}, 1 \leq i, j \leq n$ are introduced to identify the neighbor assignment for
each pair of points \( x_i, x_j \). 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:

\[
\min \ c(U, W, V; X) \\
\text{s.t.} \quad w_{id} = x_{id} \quad \forall (i, d) \in \mathcal{N}_0, \\
\quad v_{id} = x_{id} \quad \forall (i, d) \in \mathcal{N}_1, \\
\quad (U, W, V) \in \mathcal{U},
\]

where \( \mathcal{U} \) is the set of all feasible combinations \( (U, W, V) \) 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 3.2. 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. We assume the latter for simplicity of notation.

<table>
<thead>
<tr>
<th>Model</th>
<th>( c(U, W, V; X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K )-NN</td>
<td>( \sum_{i \in \mathcal{I}} \sum_{j=1}^{n} z_{ij} \left[ \sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} 1_{{v_{id} \neq v_{jd}}} \right] )</td>
</tr>
<tr>
<td>SVM</td>
<td>( \frac{1}{2}(|\beta|<em>H^2 + |\theta|<em>H^2) + C \sum</em>{i=1}^{n} \left( \sum</em>{d=1}^{p_0} (\gamma_{id} + \gamma_{id}^*) + \sum_{d=p_0+1}^{p_0+p_1} \xi_{id} \right) )</td>
</tr>
<tr>
<td>Trees</td>
<td>( T )</td>
</tr>
</tbody>
</table>

Table 3.2: Variables and cost functions for each imputation model. Variables for \( K \)-NN, SVM, and trees are defined in Sections \( 3.2.3 \), \( 3.2.4 \) and \( 3.2.5 \) respectively.

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 \texttt{opt.impute} which approximates the solution to Problem (3.1) very fast using first-order methods.
Algorithm 1 \texttt{opt.impute}

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

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

\textbf{Procedure:}

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

while $\delta > \delta_0$ do

\hspace{1em} 1. Update $U^*$ using model dependent information:

\hspace{2em} $U^* \leftarrow \arg \min_U c(U, W^{\text{old}}, V^{\text{old}}, X)$
\hspace{2em} s.t. $(U, W^{\text{old}}, V^{\text{old}}) \in \mathcal{U}$. \hspace{2em} (3.2)

\hspace{2em} 2. Update the imputation $W^*, V^*$, following one of:

\hspace{3em} 2a) block coordinate descent (BCD):

\hspace{4em} $W^*, V^* \leftarrow \arg \min_{W, V} c(U^*, W, V; X)$
\hspace{4em} s.t. $w_{id} = x_{id}$ \quad $\forall (i, d) \in \mathcal{N}_0$,
\hspace{4em} $v_{id} = x_{id}$ \quad $\forall (i, d) \in \mathcal{N}_1$,
\hspace{4em} $(U^*, W, V) \in \mathcal{U}$, \hspace{2em} (3.3)

\hspace{4em} 2b) coordinate descent (CD):

\hspace{5em} $w^*_{jr} \leftarrow \arg \min_{w_{jr}} c(U^*, W, V; X)$
\hspace{5em} s.t. $w_{id} = x_{id}$ \quad $\forall (i, d) \in \mathcal{N}_0$,
\hspace{5em} $v_{id} = x_{id}$ \quad $\forall (i, d) \in \mathcal{N}_1$,
\hspace{5em} $w_{id} = w^*_{id}$ \quad $\forall (i, d) \in \mathcal{M}_0 \setminus (j, r)$,
\hspace{5em} $v_{id} = v^*_{id}$ \quad $\forall (i, d) \in \mathcal{M}_1$, \hspace{2em} (3.4)

\hspace{5em} $v^*_{jr} \leftarrow \arg \min_{v_{jr}} c(U^*, W, V; X)$
\hspace{5em} s.t. $w_{id} = x_{id}$ \quad $\forall (i, d) \in \mathcal{N}_0$,
\hspace{5em} $v_{id} = x_{id}$ \quad $\forall (i, d) \in \mathcal{N}_1$,
\hspace{5em} $w_{id} = w^*_{id}$ \quad $\forall (i, d) \in \mathcal{M}_0$,
\hspace{5em} $v_{id} = v^*_{id}$ \quad $\forall (i, d) \in \mathcal{M}_1 \setminus (j, r)$, \hspace{2em} (3.5)

\hspace{5em} $(U^*, W, V) \in \mathcal{U}$.

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

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

end while

$X^{\text{imp}} \leftarrow [W^*; V^*]$
3.2.2 First-Order Method for the General Problem

To obtain high-quality solutions to Problem (3.1), we can use first-order methods with random warm starts. In particular, we will focus on block coordinate descent (BCD) and coordinate descent (CD) \[9\]. Algorithm 1, which we refer to as \texttt{opt.impute}, implements BCD or CD for Problem (3.1). The variables \(U, W, V,\) and \(X\) as well as the cost function \(c(\cdot)\) are summarized in Table 3.2 for K-NN, SVM, and trees. The detailed solution methods for Problems (3.2), (3.3), (3.4), and (3.5) for K-NN, SVM, and tree imputation models are described in Sections 3.2.3-3.2.5 respectively.

By construction, the objective function value strictly decreases by at least \(\delta_0\) until termination. It follows that the number of steps needed for the algorithm to terminate is \(\lceil \frac{1}{\delta_0} c(U^0, W^0, V^0; X) \rceil\), where \(W^0, V^0\) are the initialization values, \(X\) is data, and \(U^0\) is the argmin in Equation (3.2). However, the algorithm is not guaranteed to find a global minimum for Problem (3.1) \[133\].

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 \(U, W, V\) 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.

3.2.3 K-NN Based Imputation

We first define a distance metric between rows \(i\) and \(j\) as

\[
d_{ij} := \sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} 1_{\{v_{id} \neq v_{jd}\}}. \tag{3.6}
\]

Next, we introduce the binary variables:

\[
z_{ij} = \begin{cases} 
1, & \text{if row } j \text{ is among the } K\text{-nearest neighbors of row } i \\
& \text{with respect to distance metric (3.6)}, \\
0, & \text{otherwise}.
\end{cases}
\]
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:

$$
\begin{align*}
\min & \sum_{i \in \mathcal{I}} \sum_{j=1}^{n} z_{ij} \left[ \sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} 1_{\{v_{id} \neq v_{jd}\}} \right] \\
\text{s.t.} & \quad w_{id} = x_{id} \quad \forall (i, d) \in \mathcal{N}_0, \\
& \quad v_{id} = x_{id} \quad \forall (i, d) \in \mathcal{N}_1, \\
& \quad z_{ii} = 0 \quad \forall i \in \mathcal{I}, \\
& \quad \sum_{j=1}^{n} z_{ij} = K \quad \forall i \in \mathcal{I}, \\
& \quad \mathbf{Z} \in \{0, 1\}^{\vert \mathcal{I} \vert \times n}
\end{align*}
$$

(3.7)

By optimality, it follows that $z_{ij} = 1$ if and only if row $j$ is among the $K$-nearest neighbors of row $i$. Therefore, solving Problem (3.7) produces the missing value imputation which minimizes the sum of distances from each row with at least one missing entry to its $K$-nearest neighbors. Note that the relation $1_{\{v_{id} \neq v_{jd}\}}$ can be modeled with binary variables. Problem (3.7) 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 $\text{opt.knn}$. 

94
In step (1), to update the auxiliary variables $Z$, first fix all imputed values $W$, $V$. Problem (3.2) decomposes by $i \in \mathcal{I}$ into the assignment problems:

\[
\min_{z_i} \sum_{j=1}^{n} z_{ij} d_{ij}
\]
\[
s.t. \quad z_{ii} = 0,
\]
\[
\sum_{j=1}^{n} z_{ij} = K,
\]
\[
z_i \in \{0, 1\}^n
\]

The optimal solution to Problem (3.8) 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 row $i$ and set $z_{ij} = 1$ for these neighbors, $z_{ij} = 0$, otherwise.

Next, we fix $Z$ and update the imputed values $W$, $V$ using either BCD or CD. In step (2a), the BCD update, Problem (3.3) decomposes by dimension $d = 1, \ldots, p$. For each continuous dimension $d = 1, \ldots, p_0$, we consider the following quadratic optimization problem:

\[
\min_{w_d} \sum_{i \in \mathcal{I}} \sum_{j=1}^{n} z_{ij} (w_{id} - w_{jd})^2
\]
\[
s.t. \quad w_{id} = x_{id} \quad \forall (i, d) \in \mathcal{N}_0,
\]

where $w_d \in \mathbb{R}^n$ are the imputed values in the $d$th 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:

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

For each continuous dimension $d$, we have a system of equations of the form (3.9) 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 $\tilde{w} := (\tilde{w}_{1d}, \ldots, \tilde{w}_{ad})$ and the known values
are $\tilde{x} := (\tilde{x}_{(a+1)d}, \ldots, \tilde{x}_{nd})$. Then, the set of optimal imputed missing values $\tilde{w}$ is the solution to the linear system $Q\tilde{w} = R\tilde{x}$, where

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

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

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

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

$$\min_{v^d} \sum_{i \in I} \sum_{j=1}^{n} z_{ij} y_{ij}$$

s.t.  
\begin{align*}
v_{id} &= x_{id} & \forall (i, d) & \in \mathcal{N}_1, \\
v_{id} - v_{jd} & \leq y_{ij} k_d & \forall i, j, \\
v_{jd} - v_{id} & \leq y_{ij} k_d & \forall i, j, \\
y & \in \{0, 1\}^{n \times n},
\end{align*}
where \( \mathbf{v}^d \in \{1, \ldots, k_d\}^n \) are the imputed values for the \( d \)th dimension. Here, the indicator variables \( y_{ij} \) take values equal to \( 1_{\{v_{jd}\neq v_{jd}\}} \) 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:

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

Solving the above gives

\[
w_{id} = \frac{\sum_{j=1}^n z_{ij} w_{jd} + \sum_{j \in \mathcal{I}} z_{ji} w_{jd}}{K + \sum_{j \in \mathcal{I}} z_{ji}}.
\] (3.10)

We can interpret the missing value imputation (3.10) as a weighted average of the \( K \) nearest neighbors of \( x_i \), along with all points \( x_j \) which include \( x_i \) as a neighbor. Similarly, each categorical variable \( v_{id}, (i, d) \in \mathcal{M}_1 \) is imputed as the minimizer of the following:

\[
\min_{v_{id}} \sum_{j=1}^n z_{ij} 1_{\{v_{jd}\neq v_{jd}\}} + \sum_{j \in \mathcal{I}} z_{ji} 1_{\{v_{jd}\neq v_{jd}\}}.
\]

The solution is

\[
v_{id} = \text{mode}(\{v_{jd} : z_{ij} = 1\}, \{v_{jd} : z_{ji} = 1\}).
\]

Here, we set \( v_{id} \) to be the highest frequency category among the \( K \) nearest neighbors of \( x_i \), along with all points \( x_j \) which include \( x_i \) 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 \( V \) in BCD because it is much faster computationally.

### 3.2.4 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 \( \tilde{\mathbf{v}}_i \in \{-1, 1\}^{p_2} \) to be a dummy encoded representation of \( \mathbf{v}_i \), where \( p_2 = \sum_{d=p_0+1}^{p_1} k_d - p_1 \). Let \( \tilde{v}_{id}^{fixed}, (i, d) \in \mathcal{N}_2 \) be the known dummy encoded values. For each continuous
feature \( d \in \{1, \ldots, p_0\} \), let \((\beta_d, \beta_{d0}) \in \mathbb{R}^{p_0+p_2+1}\) be the coefficients for an SVM regression model regressing feature \( d \) on the other features with the dummy encoding. Let \((\theta_d, \theta_{d0}) \in \mathbb{R}^{p_0+p_2+1}\) 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 by [47], using parameters of the form \( M \in \mathbb{R}^{k_d \times (p_0+p_2+1)} \) 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 \( \tilde{v}_{id}, (i, d) \in \mathcal{M}_2 \). 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:

\[
\begin{align*}
\min & \quad \frac{1}{2} \left( \|\theta\|^2 + \|\beta\|^2 \right) + C \left( \sum_{i=1}^{n} \sum_{d=1}^{p_0} (\gamma_{id} + \gamma_{id}^*) + \sum_{i=1}^{n} \sum_{d=p_0+1}^{p_0+p_1} \xi_{id} \right) \\
\text{s.t.} & \quad x_{id} = w_{id} \quad \forall (i, d) \in \mathcal{N}_0, \\
& \quad \tilde{v}_{id} = \tilde{v}_{id}^{fixed} \quad \forall (i, d) \in \mathcal{N}_2, \\
& \quad \beta_{dd} = 0 \quad \forall d, \\
& \quad \theta_{dd} = 0 \quad \forall d, \\
& \quad \gamma_{id} \geq w_{id} - \left( \beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_{d0} \right) - \epsilon \quad \forall i, d, \\
& \quad \gamma_{id}^* \geq \left( \beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_{d0} \right) - w_{id} - \epsilon \quad \forall i, d, \\
& \quad \xi_{id} \geq 1 - \tilde{v}_{id} \left( \theta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \theta_{d0} \right) \quad \forall i, d, \\
& \quad \gamma_{id} \geq 0 \quad \forall i, d, \\
& \quad \gamma_{id}^* \geq 0 \quad \forall i, d, \\
& \quad \xi_{id} \geq 0 \quad \forall i, d, \\
& \quad \tilde{v} \in \{-1,1\}^{n \times p_2}.
\end{align*}
\]
This formulation is based upon SVM with a linear kernel; however we can extend Problem (3.11) to arbitrary kernels, including the multi-class cases, using the modified objective function

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

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

Another important aspect of Problem (3.11) 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 $\tilde{v}_{id}$, the contribution to the objective function scales linearly as $(\beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_{d0})$ if $d$ is continuous or scales linearly as $(\theta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \theta_{d0})$ 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 $\text{opt.svm}$.

In step (1), we fix the imputed values $W, V$ and update the auxiliary variables $[\beta, \beta_0, \theta, \theta_0]$. Independent of the choice of kernel, Problem (3.2) 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 $\beta_d, \beta_0$ by solving

$$
\min \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} (\gamma_{id} + \gamma_{id}^*) \\
\text{s.t. } \beta_{dd} = 0 \\
\gamma_{id} \geq w_{id} - (\beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_0) - \epsilon \quad \forall i, \\
\gamma_{id}^* \geq (\beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_0) - w_{id} - \epsilon \quad \forall i, \\
\gamma_{id} \geq 0 \quad \forall i, \\
\gamma_{id}^* \geq 0 \quad \forall i.
$$

(3.12)

Similarly, for each dummy feature $d \in \{p_0 + 1, \ldots, p_0 + p_2\}$, we update $\theta_d, \theta_0$ by solving

$$
\min \frac{1}{2} \|\theta\|^2 + C \sum_{i=1}^{n} \xi_{id} \\
\text{s.t. } \theta_{dd} = 0 \\
\xi_{id} \geq 1 - \tilde{v}_{id}(\theta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \theta_0) \quad \forall i, \\
\xi_{id} \geq 0 \quad \forall i.
$$

(3.13)

Taking the Lagrangian duals, both Problems (3.12) and (3.13) can be reformulated as quadratic optimization problems which can be solved efficiently [46].

Next, we fix the auxiliary variables $[\beta, \beta_0, \theta, \theta_0]$ and update the imputed values $W, V$ using BCD or CD. In step (2a), Problem (3.2) decomposes by observation $i$ into $n$ nonlinear
integer optimization problems. For each $i$ we solve

$$\min_{w_i, \bar{v}_i} \sum_{d=1}^{p_0} (\gamma_{id} + \gamma^*_{id}) + \sum_{d=p_0+1}^{p_0+p_1} \xi_{id}$$

s.t. $x_{id} = w_{id} \quad \forall (i, d) \in \mathcal{N}_0,$

$$\gamma_{id} \geq w_{id} - (\beta_d^T \begin{bmatrix} w_i \\ \bar{v}_i \end{bmatrix} + \beta_{d0}) - \epsilon \quad \forall d,$$

$$\gamma^*_{id} \geq (\beta_d^T \begin{bmatrix} w_i \\ \bar{v}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon \quad \forall d,$$

(3.14)

$$\xi_{id} \geq 1 - \bar{v}_{id}(\theta_d^T \begin{bmatrix} w_i \\ \bar{v}_i \end{bmatrix} + \theta_{d0}) \quad \forall d,$$

$$\gamma_{id} \geq 0 \quad \forall d,$$

$$\gamma^*_{id} \geq 0 \quad \forall d,$$

$$\xi_{id} \geq 0 \quad \forall d,$$

where $(w_i, \bar{v}_i) \in \mathbb{R}^{p_0} \times \{-1, 1\}^{p_2}$ is the imputation for observation $x_i$. Note that if all features are continuous, Problem (3.14) 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 (3.14).

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:

$$\min_{w_{id}} \sum_{d=1}^{p_0} (\gamma_{id} + \gamma^*_{id}) + \sum_{d=p_0+1}^{p_0+p_1} \xi_{id}$$

s.t. 

$$\gamma_{id} \geq w_{id} - (\beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_{d0}) - \epsilon \quad \forall d,$$

$$\gamma^*_{id} \geq (\beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon \quad \forall d,$$

$$\xi_{id} \geq 1 - \tilde{v}_{id}(\theta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \theta_{d0}) \quad \forall d,$$

$$\gamma_{id} \geq 0 \quad \forall d,$$

$$\gamma^*_{id} \geq 0 \quad \forall d,$$

$$\xi_{id} \geq 0 \quad \forall d.$$

We update $\tilde{v}_{id}, (i, d) \notin N_2$ by solving the binary optimization problem:

$$\min_{\tilde{v}_{id} \in \{-1,1\}} \sum_{i=1}^{n} \sum_{d=1}^{p_0} \left( \max\{w_{id} - (\beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_{d0}) - \epsilon, 0\} + \max\{(\beta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon, 0\} \right) + \sum_{i=1}^{n} \sum_{d=1}^{p_2} (1 - \tilde{v}_{id}(\theta_d^T \begin{bmatrix} w_i \\ \tilde{v}_i \end{bmatrix} + \theta_{d0})).$$

### 3.2.5 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 $x_i$. 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 $x_i$.

For general prediction tasks, we can use greedy [32] or globally optimal [12] 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 $T^d \in \{0,1\}^{n \times n}$ denote the set of indicator variables

$$t^d_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same leaf node of the decision tree for dimension } d, \\ 0, & \text{otherwise.} \end{cases}$$

Let $(T^d, W, V) \in \mathcal{T}^d$ denote the set of optimal decision tree constraints for dimension $d$ as described in [12]. We consider the following optimization problem:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \left[ \sum_{d=1}^{p_0} t^d_{ij} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} t^d_{ij} 1\{v_{id} \neq v_{jd}\} \right]$$

s.t. $w_{id} = x_{id}$ $\forall (i, d) \in \mathcal{N}_0,$

$v_{id} = x_{id}$ $\forall (i, d) \in \mathcal{N}_1,$

$(T^d, W, V) \in \mathcal{T}^d$ $\forall d.$  \hspace{1cm} (3.15)

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

\texttt{opt.tree}

In step 1, we fix the imputed values $W, V$ and update the decision tree variables $T$. For each continuous feature, we fit a regression tree to predict $w^d$ based upon the other features. Similarly, for each categorical feature, we fit a classification tree to predict $v^d$ 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 \( c(T, W, V; X) \) is not guaranteed to be monotonically decreasing over the course of the algorithm.

Next, we fix \( T \) and update the imputed values \( W, V \) using BCD or CD. In step (2a), Problem (3.3) 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:

\[
\min_{w^d} \sum_{i=1}^{n} \sum_{j=1}^{n} t^d_{ij} (w_{id} - w_{jd})^2
\]

\[
\text{s.t. } w_{id} = x_{id} \quad \forall (i, d) \in N_0,
\]

where \( w^d \in \mathbb{R}^n \) are the imputed values in the \( d \)th dimension. This is a quadratic optimization problem with an explicit optimum. For each \( w_{id}, (i, d) \in M_0 \), an optimal solution is

\[
w_{id} = \begin{cases} 
\frac{\sum_{(j,d) \in N^d_0} t^d_{ij} x_{jd}}{\sum_{(j,d) \in N^d_0} t^d_{ij}}, & \text{if } \sum_{(j,d) \in N^d_0} t^d_{ij} \geq 1, \\
\frac{1}{|N^d_0|} \sum_{(j,d) \in N^d_0} x_{jd}, & \text{otherwise},
\end{cases}
\]

where \( N^d_0 := \{(i, r) \in N_0 : r = d\} \). 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., \( \sum_{(j,d) \in N^d_0} t^d_{ij} = 0 \), 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:

\[
\min_{v^d} \sum_{i=1}^{n} \sum_{j=1}^{n} t^d_{ij} \mathbb{1}_{\{v_{id} \neq v_{jd}\}}
\]

\[
\text{s.t. } v_{id} = x_{id} \quad \forall (i, d) \in N_1,
\]

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

\[ v_{id} = \begin{cases} 
\text{mode}\left( \{ x_{jd} : t_{ij}^d = 1, (j, d) \in N_1 \} \right) & \text{if } |\{ x_{jd} : t_{ij}^d = 1, (j, d) \in N_1 \}| \geq 1, \\
\text{mode}\left( \{ x_{jd} : (j, d) \in N_1 \} \right) & \text{otherwise}.
\end{cases} \]

In step (2b), we update the missing imputed values one at a time, which results in slightly different closed form solutions for \( w_{id}, (i, d) \in M_0 \) and \( v_{id}, (i, d) \in M_1 \). First, we update the continuous variables \( w_{id}, (i, d) \in M_0 \) by solving:

\[
\min_{w_{id}} 2 \sum_{j=1}^{n} t_{ij}^d (w_{id} - w_{jd})^2. \tag{3.16}
\]

An optimal solution to Problem (3.16) is

\[ w_{id} = \begin{cases} 
\frac{\sum_{j \neq i} t_{ij}^d w_{jd}}{\sum_{j \neq i} t_{ij}^d}, & \text{if } \sum_{j \neq i} t_{ij}^d \geq 1, \\
\frac{1}{|N_0^d|} \sum_{(j,d) \in N_0^d} x_{jd}, & \text{otherwise}.
\end{cases} \]

Next, we update the categorical variables \( v_{id}, (i, d) \in M_1 \) one at a time by solving:

\[
\min_{v_{id}} 2 \sum_{j=1}^{n} t_{ij}^d 1_{\{v_{id} \neq v_{jd}\}}. \tag{3.17}
\]

An optimal solution to Problem (3.17) is

\[ v_{id} = \begin{cases} 
\text{mode}\left( \{ v_{jd} : t_{ij}^d = 1 \} \right), & \text{if } |\{ v_{jd} : t_{ij}^d = 1 \}| \geq 1, \\
\text{mode}\left( \{ x_{jd} : (j, d) \in N_1 \} \right), & \text{otherwise}.
\end{cases} \]

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

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

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

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

For each imputation method, the combination of hyperparameters that achieves the lowest MAE in validation (or RMSE) is selected, and the $X$ 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 3.3. 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 the best method out of: mean, pmm, bpca, knn, and iknn that will be later used in computational comparisons (see Section 3.3.1 for descriptions of these individual methods).
<table>
<thead>
<tr>
<th>Method</th>
<th>Hyperparameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-NN</td>
<td>$K$</td>
</tr>
<tr>
<td>SVM</td>
<td>$C$, $\sigma^2$</td>
</tr>
<tr>
<td>Trees</td>
<td>$cp$</td>
</tr>
</tbody>
</table>

Table 3.3: Hyperparameters tuned via the model selection procedure outlined in Section 3.2.6. $\sigma^2$ is a parameter in the radial basis function kernel, $K(x_i, x_j) = \exp\left(\frac{\|x_i - x_j\|}{\sigma^2}\right)$. $cp$ is a complexity parameter related to the depth of the decision tree.

### 3.2.7 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 [82]. 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.

### 3.3 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.

#### 3.3.1 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 3.10.

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 $X$ 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 val-
ues 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 \cite{125}. Implemented using the MICE package in R.

3. **Bayesian PCA (bpca):** A missing data estimation method based on Bayesian principal component analysis \cite{99}. 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 \cite{31, 41}.

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 \cite{3.7}. 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 \cite{3.11} using a radial basis function kernel. For continuous variables, we use $\varepsilon$-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 (\texttt{opt.tree}): This method solves the optimal decision-tree problem \((3.15)\). 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 \([12]\). We run coordinate descent for the imputation problems. The implementation was in \texttt{Julia} using the packages \texttt{glmnet} and \texttt{OptimalTrees}.

In addition, we consider two composite methods: \texttt{opt.cv}, which selects the best method from \texttt{opt.knn}, \texttt{opt.svm}, and \texttt{opt.tree}; and \texttt{benchmark.cv}, which selects the best method from \texttt{mean}, \texttt{pmm}, \texttt{b pca}, \texttt{knn}, and \texttt{iknn}. These composite methods use the cross-validation procedure described in Section \(3.2.6\). 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 \(3.2.6\). For each of the \texttt{opt.impute} methods, we also record and present the convergence in objective value and MAE to show the progress over the iterations.

**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 \(3.4\). 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
Mechanism of Missing Data | Assumption
--- | ---
Missing Completely at Random (MCAR) | \( f(\mathcal{M} | X^{\text{obs}}, X^{\text{miss}}) = f(\mathcal{M}) \)
Missing at Random (MAR) | \( f(\mathcal{M} | X^{\text{obs}}, X^{\text{miss}}) = f(\mathcal{M} | X^{\text{obs}}) \)
Not Missing at Random (NMAR) | \( f(\mathcal{M} | X^{\text{obs}}, X^{\text{miss}}) \) is a function of \( X^{\text{miss}} \)

Table 3.4: Statistical assumptions of mechanisms used to generate patterns of missing data \( \mathcal{M} \) for data set \( X \). Here, we suppose that \( f \) is the underlying density of the missing pattern, and \( X^{\text{obs}}, X^{\text{miss}} \) are the observed and missing components of the data set, respectively.

To generate MCAR patterns of missing data, we randomly sample a subset of the entries in \( X \) 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 \( X^d \) 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 \( Y \in \mathbb{R}^n \) and \( Y \in \{0,1\}^n \) respectively. We evaluate both single and multiple imputation methods in these experiments.

For single imputation, we consider \texttt{opt.cv} and \texttt{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 \( X \) 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.

3.3.2 **Results**

We run the methods on 84 data sets from the UCI Machine Learning Repository. These data sets range in size from $n = 23$ to 5,875 observations and dimension $p = 2$ to 124. 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 3-1 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 $K$-NN 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 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 \texttt{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 \texttt{opt.svm} and \texttt{opt.tree}.

**Imputation Accuracy**

The imputation accuracy for each data set is presented in Table 3.10 for the scenario in which 30\% of the entries are missing, assuming MCAR. We compare the benchmark ones and each individual \texttt{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 \texttt{opt.impute} methods obtains the lowest MAE in 76.2\% of the data sets, followed by \texttt{iknn} and \texttt{bpcal} imputation methods with 9 and 4 wins each. Comparatively, \texttt{mean}, \texttt{knn}, and \texttt{pmm} impute have the weakest performances. Among the \texttt{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 3.2 using \texttt{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, \texttt{mean} and \texttt{pmm} are generally among the weaker ones. When MAE is the metric, the heuristic
method \texttt{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), \texttt{bpca} improves in its performance when RMSE is the metric of evaluation, but still not as strong as \texttt{opt.cv}.

In Figure 3-3, 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, \texttt{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 \texttt{opt.cv} imputations is 0.100, a reduction of 14.9\% from the average MAE of 0.118 obtained by the best benchmark method \texttt{knn}. As missing percentages increase, \texttt{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 \texttt{knn}. The performance of \texttt{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.

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 3.5 presents the statistical comparisons between each \texttt{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 \texttt{opt.cv} against the benchmark methods, our proposed cross-validated method achieves statistically significant lower rank and lower MAE compared to each benchmark. For each individual \texttt{opt.impute} method, with the exception of \texttt{opt.svm} against heuristic \texttt{iknn}, the \texttt{opt.impute} one has statistically significant lower rank than every benchmark. The decrease in MAE is still statistically significant when \texttt{mean}, \texttt{bpca}, and \texttt{pmm} are comparators, but no longer statistically significant when compared to \texttt{knn} or \texttt{iknn}. This suggests that each of the proposed methods holds its own against most benchmark ones, especially under rank comparisons, but the cross-validation procedure adds another layer of improvement in imputation quality.
Figure 3-2: Number of data sets in which each missing data imputation method achieves lowest mean absolute error (MAE) or root mean squared error (RMSE) from true value, with ties included. 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 3.5: Pairwise Wilcoxon signed-rank tests and t-tests between **opt.impute** and benchmark methods, with the \( p \)-values adjusted for multiple comparisons.

Finally, we compare against the same cross-validated procedure introduced in Section 3.2.6 applied on all the benchmark methods (**benchmark.cv**) with results in Figure 3-2b. 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 data sets 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 3.6, 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 5,875 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
Figure 3-3: Mean absolute error (MAE) and root mean squared error (RMSE) across 84 data sets for each imputation method, comparing \texttt{opt.cv} against all benchmark methods and against the cross-validated best benchmark method, \texttt{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.
Table 3.6: 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.

sets, the downstream tasks become more difficult as the missing data percentage increases.

In Figure 3-4, we show how the imputation method chosen impacts the performance for downstream tasks, across different data sets and different missing data percentages. In Tables 3.7 and 3.8, 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 the improvement of opt.cv over the best cross-validated benchmark method is statistically significant 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 the improvement of opt.mi over mice is statistically significant 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.

<table>
<thead>
<tr>
<th>Downstream Task</th>
<th>Name</th>
<th>$(n,p)$</th>
<th>Baseline Accuracy or $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classification</strong></td>
<td>climate-model-crashes</td>
<td>(540, 18)</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>connectionist-bench</td>
<td>(990, 10)</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>ecoli</td>
<td>(336, 8)</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>iris</td>
<td>(150, 4)</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>pima-indians-diabetes</td>
<td>(768, 8)</td>
<td>0.77</td>
</tr>
<tr>
<td><strong>Regression</strong></td>
<td>abalone</td>
<td>(4177, 7)</td>
<td>0.51</td>
</tr>
<tr>
<td></td>
<td>auto-mpg</td>
<td>(392, 8)</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>housing</td>
<td>(506, 13)</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>parkinsons-telemonitoring-total</td>
<td>(5875, 16)</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>wine-quality-white</td>
<td>(4898, 11)</td>
<td>0.27</td>
</tr>
</tbody>
</table>
Overall, these results suggest that \texttt{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 \texttt{opt.impute}. We observe that the improvement of \texttt{opt.mi} over \texttt{opt.cv} is statistically significant 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 \texttt{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 of neighbors $K$ in $K$-NN and the trade-off parameter $C$ for SVM. It is also affected by the number
(a) Average out-of-sample accuracy values with standard errors of Optimal Trees and SVM models.

(b) Average out-of-sample $R^2$ values with standard errors of SVR and LASSO models.

Figure 3-4: Average out-of-sample performance of downstream models trained on data imputed via `opt.impute` and benchmark methods across a sample of classification and regression problems and a range of missing data percentages. Multiple and single imputation methods are solid and dotted lines respectively.
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 3-5 shows the relationship between the hyperparameters and MAE for various data sets and missing patterns. For \texttt{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 \texttt{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 \texttt{opt.impute} methods are relatively robust even if their hyperparameters are not known exactly.

For \texttt{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
<table>
<thead>
<tr>
<th>((n,p))</th>
<th>Missing %</th>
<th>Benchmark</th>
<th>(\text{opt.impute})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>bPCA</td>
<td>knn</td>
</tr>
<tr>
<td>(150, 4)</td>
<td>10</td>
<td>0.802</td>
<td>0.088</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>1.717</td>
<td>0.446</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>1.875</td>
<td>0.736</td>
</tr>
<tr>
<td>(1372, 4)</td>
<td>10</td>
<td>2.622</td>
<td>2.552</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>14.058</td>
<td>14.914</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>17.820</td>
<td>16.889</td>
</tr>
<tr>
<td>(360, 90)</td>
<td>10</td>
<td>2.624</td>
<td>0.088</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>3.423</td>
<td>0.446</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>1.892</td>
<td>0.736</td>
</tr>
<tr>
<td>(5644, 76)</td>
<td>10</td>
<td>26.634</td>
<td>387.386</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>46.726</td>
<td>8.134</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>63.556</td>
<td>10.155</td>
</tr>
<tr>
<td>(245057, 3)</td>
<td>10</td>
<td>392.310</td>
<td>1144.120</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>450.584</td>
<td>1380.138</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>615.037</td>
<td>2503.464</td>
</tr>
<tr>
<td>(1080, 856)</td>
<td>10</td>
<td>30.310</td>
<td>13.038</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>58.205</td>
<td>13.970</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>126.059</td>
<td>14.361</td>
</tr>
</tbody>
</table>

Table 3.9: Computational time comparison of benchmark and \textit{opt.impute} imputation methods on UCI data sets of varying sizes. Blank entries indicate that the method failed to converge with the 4 hour time limit.

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 \textit{opt.impute} methods, we report the running times for \texttt{mean} warm starts, as multiple warm starts can be trivially parallelized. The results are shown below in Table 3.9.

Mean imputation is almost instantaneous and is therefore not presented in the table. For small-scale problems on the \texttt{iris} data set, all imputation methods finish quickly. As the data dimension \(p\) increases (for example, in the \texttt{libras-movement} data set), most \textit{opt.impute} methods scale better than the \texttt{pmm} method. As the sample size \(n\) increases, \textit{opt.knn.CD} also scales better than \texttt{pmm}, as seen in \texttt{banknote-authentication} and \texttt{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 \( X \) to update \( Z \) 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.

### 3.4 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 84 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 a statistically 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 produce 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 work, 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 (3.1) fast and accurately for any of the three examples of non-convex, non-linear cost functions $c(U, W, V; X)$ remains an open question.

### 3.5 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, \texttt{opt.impute}, which finds high quality solutions to this problem using fast first-order methods. Through extensive computational experiments on 84 data sets from the UCI Machine Learning Repository, we show that \texttt{opt.impute} yields statistically 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 improves over state-of-the-art methods across a broad range of missing data scenarios.
Table 3.10: Mean absolute errors of imputation methods on 84 data sets from the UCI Machine Learning repository with 30% missing values. The lowest MAE for each data set is indicated in bold.
Table 3.10: (cont.) Mean absolute errors of imputation methods on 84 data sets from the UCI Machine Learning repository with 30% missing values. The lowest MAE for each data set is indicated in **bold**.
<table>
<thead>
<tr>
<th>Name</th>
<th>n</th>
<th>p₀</th>
<th>p₁</th>
<th>mean</th>
<th>pm</th>
<th>bpc</th>
<th>kn</th>
<th>ik</th>
<th>knn</th>
<th>opt.impute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Benchmark</td>
</tr>
<tr>
<td>triazines</td>
<td>186</td>
<td>60</td>
<td>0</td>
<td>0.1574</td>
<td>0.0667</td>
<td>0.1184</td>
<td>0.0503</td>
<td>0.0708</td>
<td><strong>0.0454</strong></td>
<td>0.0892</td>
</tr>
<tr>
<td>tv-sales</td>
<td>31</td>
<td>8</td>
<td>0</td>
<td>0.2073</td>
<td>0.1949</td>
<td>0.1808</td>
<td>0.1934</td>
<td><strong>0.1729</strong></td>
<td>0.1952</td>
<td>0.1731</td>
</tr>
<tr>
<td>vote-for-clinton</td>
<td>2704</td>
<td>9</td>
<td>0</td>
<td>0.0644</td>
<td>0.0715</td>
<td>0.0538</td>
<td>0.0676</td>
<td>0.0552</td>
<td><strong>0.0523</strong></td>
<td>0.0633</td>
</tr>
<tr>
<td>wall-following-robot-2</td>
<td>5456</td>
<td>2</td>
<td>0</td>
<td>0.0721</td>
<td>0.0955</td>
<td>0.0721</td>
<td>0.0754</td>
<td>0.0720</td>
<td>0.0792</td>
<td>0.0847</td>
</tr>
<tr>
<td>wall-following-robot-24</td>
<td>5456</td>
<td>4</td>
<td>0</td>
<td>0.0917</td>
<td>0.1172</td>
<td>0.0917</td>
<td>0.0886</td>
<td>0.0872</td>
<td><strong>0.0862</strong></td>
<td>0.0946</td>
</tr>
<tr>
<td>wiki4he</td>
<td>176</td>
<td>0</td>
<td>44</td>
<td>0.2200</td>
<td>0.2234</td>
<td>0.1857</td>
<td>0.1872</td>
<td>0.1968</td>
<td>0.1777</td>
<td><strong>0.1731</strong></td>
</tr>
<tr>
<td>wine-quality-red</td>
<td>1599</td>
<td>11</td>
<td>0</td>
<td>0.0976</td>
<td>0.0945</td>
<td>0.0761</td>
<td>0.0796</td>
<td>0.0744</td>
<td><strong>0.0683</strong></td>
<td>0.0757</td>
</tr>
<tr>
<td>wine-quality-white</td>
<td>4898</td>
<td>11</td>
<td>0</td>
<td>0.0756</td>
<td>0.0782</td>
<td>0.0668</td>
<td>0.0771</td>
<td>0.0645</td>
<td>0.0598</td>
<td>0.0676</td>
</tr>
<tr>
<td>wine</td>
<td>178</td>
<td>13</td>
<td>0</td>
<td>0.1680</td>
<td>0.1537</td>
<td>0.1203</td>
<td>0.1184</td>
<td>0.1144</td>
<td><strong>0.1091</strong></td>
<td>0.1105</td>
</tr>
<tr>
<td>yacht-hydrodynamics</td>
<td>308</td>
<td>6</td>
<td>0</td>
<td>0.2102</td>
<td>0.1991</td>
<td>0.2088</td>
<td>0.1858</td>
<td>0.1861</td>
<td>0.1866</td>
<td>0.1867</td>
</tr>
<tr>
<td>yeast</td>
<td>1484</td>
<td>8</td>
<td>0</td>
<td>0.0721</td>
<td>0.0917</td>
<td>0.0689</td>
<td>0.0740</td>
<td><strong>0.0671</strong></td>
<td>0.0683</td>
<td>0.0928</td>
</tr>
<tr>
<td>zoo</td>
<td>101</td>
<td>1</td>
<td>15</td>
<td>0.2892</td>
<td>0.2832</td>
<td>0.1835</td>
<td>0.1518</td>
<td>0.2860</td>
<td>0.1502</td>
<td>0.3637</td>
</tr>
</tbody>
</table>

Table 3.10: (cont.) Mean absolute errors of imputation methods on 84 data sets from the UCI Machine Learning repository with 30% missing values. The lowest MAE for each data set is indicated in **bold**.
Chapter 4

Imputation of Clinical Covariates in Time Series

This work has been submitted for publication, with co-authors Dimitris Bertsimas and Agni Orfanoudaki [20].

Missing data is a common problem in longitudinal data sets which include multiple instances of the same individual observed at different points in time. Building upon the imputation framework proposed in Chapter 3, we introduce a new approach MedImpute for imputing missing clinical covariates in multivariate panel data. This approach integrates patient specific information into an optimization formulation that can be adjusted for different imputation models. We present the formulation for a $K$-nearest neighbors model and derive a corresponding scalable first-order method $\text{med.knn}$. Our algorithm provides imputations for data sets with both continuous and categorical features and observations occurring at arbitrary points in time. We test its performance on two distinctive real-world clinical data sets under different missing data patterns, and we show that $\text{med.knn}$ leads to significant improvements in both imputation accuracy and downstream model AUC compared to state-of-the-art methods. On longitudinal data from the Framingham Heart Study, which includes participants with 10 or more observations recorded regularly every 2 years, $\text{med.knn}$ imputation leads to an increase of 6.2% in AUC for predicting 10-year risk of stroke.
with 50% data missing. On electronic health record data for breast cancer patients from the Dana Farber Cancer Institute, where observations are recorded irregularly and more than half of the participants have only a single observation in the data set, med.knn imputation leads to an increase of 0.5% in AUC for predicting 60-day risk of mortality with 50% data missing. These improvements are relative to the best of the benchmark imputation methods considered.

4.1 Introduction

Machine learning applied to healthcare data can generate actionable insights ranging from predicting the onset of disease to streamlining hospital operations. Statistical models that leverage the variety and richness of clinical data are still relatively rare and offer an exciting avenue for further research [36]. As an increasing amount of information becomes available the medical field expects machine learning to become an indispensable tool for clinicians [100].

This information will come from various clinical and epidemiological sources. Claims records, clinical trials, and data from longitudinal studies have been an invaluable resource for medical research over the past decades. In many of these data sets, data from individual subjects is gathered over time via continuous or repeated monitoring of both risk factors and health outcomes. For example, longitudinal cohort studies are used to discover relationships between exposures of interest and long term health effects including adverse events and chronic disease. By design, these studies mitigate recall bias in participants by collecting data prospectively and prior to knowledge of a possible subsequent event [40].

Another valuable source of clinical data are Electronic Health Records (EHR). Over the past years, widespread uptake of EHR has generated massive data sets that contain quantitative, qualitative, and transactional data [117]. Their hospital adoption has skyrocketed in part due to the Health Information Technology for Economic and Clinical Health (HITECH) Act of 2009, which provided $30 billion in incentives for hospitals and physician practices to adopt EHR systems [29]. While primarily designed for archiving patient information and
performing administrative healthcare tasks, many researchers have found secondary use of these records for various clinical informatics applications [110]. Because heterogeneous labs, measurements, and notes are recorded for patients during each visit, EHR data has a rich and complex structure with time series information.

However, it is algorithms and not merely data sets that will prove transformative for the medical field [100]. To make progress, we need to develop new statistical tools tailored to clinical applications which address the challenges and leverage common structure encountered in healthcare data. One of the most important issues is the ubiquitous presence of missing time series data [103], particularly for variables requiring complex, time-sensitive, or resource-intensive procedures to collect. There are many reasons for “missingness”, including missed study visits, patients lost to follow-up, missing information in source documents, lack of availability (e.g., laboratory tests that were not performed), and clinical scenarios preventing collection of certain variables (e.g., missing coma scale data in sedated patients) [42]. Thus, creating a consistent data set for individuals over multiple visits even at the same healthcare organization for a fixed set of covariates remains a challenge. Even in longitudinal studies, where a set of covariates is collected over time, missing data are pervasive and complete ascertainment of all variables is rare [76].

The presence of missing data poses considerable challenges in the analyses and interpretation of clinical investigations’ results [132], potentially weakening their validity and leading to biased inferences. Their presence may complicate interpretation or even invalidate an otherwise important study [130]. Many methods commonly used for handling missing values during data analysis can yield biased results, decrease study power, or lead to underestimates of uncertainty, all reducing the chance of drawing valid conclusions [42]. As many statistical models and machine learning algorithms rely on complete data sets, it is key to handle the missing data appropriately.

4.1.1 Review of Methods for Handling Missing Values

Excluding observations that contain missing values has been a standard practice for clinical research, primarily due to the lack of interpretable, accurate machine learning methods
that can be easily applied by medical researchers [114, 68]. Unsurprisingly, complete case analysis may suffer from severe bias and the reduced sample size results in lower study power [42]. Recent advances in machine learning have allowed missing values to be accurately imputed prior to running statistical analyses on the complete data set. The benefit of the latter approach is 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. In healthcare settings, often times those data sets contain numerous visits of the same person corresponding to various patterns of missing data. This special structure challenges state-of-the-art missing data methods which do not consider the connection of multiple observations to the same individual [43].

A variety of machine learning approaches have been introduced in the literature to impute missing values ignoring the potential dependency between observations of the same individual. The simplest approach is the mean imputation that uses the mean of the observed values to replace those missing for the same covariate [91]. However, mean imputation underestimates the variance, ignores the correlation between the features leading to poor imputation outcomes.

Another common method called bPCA uses the singular value decomposition (SVD) of the data matrix and information from a Bayesian prior distribution on the model parameters to impute missing values. This method outperforms basic SVD methods [99]. In cases where the level of missing data is above 30%, we have found that this method reduces to mean imputation, leading to similar biases.

Joint modeling assumes the existence of a joint distribution on the entire data set and a parametric density function on the data given model parameters. Current implementations of the method estimate the model parameters using an Expectation-Maximization (EM) approach in order to maximize the likelihood function. One widely used software package which implements this approach, Amelia 2, assumes that data are drawn from a multivariate normal distribution [63]. In practice, healthcare data typically violate this condition.

Recent review articles indicate that single imputation methods can lead to seriously misleading results and advise us to consider multiple imputation [68, 91]. This approach,
implemented in the software package mice, allows for uncertainty about the missing data by creating several different plausible imputed data sets and appropriately combining results obtained from each of them [109]. Multiple imputation entails two stages: (1) generating replacement values for missing data and repeating this procedure many times, resulting in many data sets with replaced missing information, and (2) analyzing the many imputed data sets and combining the results [101]. As a result, multiple imputation methods are slower and require pooling results, which may not be appropriate for certain applications. For example, in clinical applications, where the interpretability of the underlying model matters, a single imputed data set and simple predictive model may be preferred.

Most recently, Bertsimas et al. [22] proposed a general optimization framework with a predictive model-based cost function that can explicitly handle both continuous and categorical variables and can be used to generate single, as well as multiple, imputations. This optimization perspective has led to new scalable algorithms for more accurate data imputation. We describe this method OptImpute in detail in Chapter 3 which we use as a foundation for the imputation method proposed in this chapter.

The algorithms above are not tailored to multivariate time series data sets despite the fact that covariates may be strongly correlated over time [81]. Preliminary work has been done demonstrating their performance in that setting [140]. Recurrent Neural Network approaches have also been employed to handle missing values in time series among the covariates for a particular prediction task [81] [43]. While the above methods have shown promising results in this direction, to date there is not an available package to test their performance that can be tested with other data sets.

4.1.2 Contributions

Given multivariate time series data, we develop a novel imputation method that utilizes optimization and machine learning techniques and outperforms state-of-the-art algorithms. Our contributions are as follows:

1. We formulate the problem of missing data imputation with time series information under the MedImpute framework, extending the OptImpute framework proposed in
Our approach can be adjusted to account for different imputation models based on predictive methods such as $K$-NN, SVM, and trees. We focus on a $K$-NN formulation to solve the problem and derive a corresponding fast first-order algorithm \texttt{med.knn}. This method provides imputations for data sets with both continuous and categorical features and observations occurring at arbitrary points in time.

2. We show that \texttt{med.knn} outperforms state-of-the-art imputation methods on two real-world sets of data with direct clinical implications. We consider the Framingham Heart Study (FHS), a longitudinal data set with rich time series data recorded at regular time intervals, and electronic health record (EHR) data from the Dana Farber Cancer Institute, which is less structured and more sparse time series data. On the FHS data set with 50% data missing, \texttt{med.knn} decreases the imputation error by 32.9% and improves the AUC of a logistic regression model for predicting 10-year risk of stroke from 0.773 to 0.837. On the Dana Farber EHR data set with 50% data missing, \texttt{med.knn} decreases the imputation error by 14.5% and improves the AUC of a logistic regression model for predicting 60-day risk of mortality from 0.896 to 0.901. These improvements are relative to the best of the comparator methods among mean, bPCA, \texttt{mice}, and \texttt{opt.knn}, which are described in Section 4.3.

3. We design and run informative experiments on the FHS and Dana Farber EHR data sets that assess the performance of our methods varying the percentage of missing data, number of observations per individual, and mechanism of missing data. We demonstrate that \texttt{med.knn} gives the highest AUC and lowest imputation error as we vary the missing percentage from 10% to 50%. As we increase the number of observations per individual, the relative performance of \texttt{med.knn} improves until it is the best method for both data sets. Finally, we run experiments changing the missing data mechanism from missing completely at random (MCAR) to not missing at random (NMAR). These NMAR experiments demonstrate that \texttt{med.knn} performs well on missing patterns commonly encountered in practice for both longitudinal studies and EHR data.

The structure of this chapter is as follows. In Section 4.2, we describe our framework for
imputation of clinical covariates in time series and proposed method \texttt{med.knn}. In Section 4.3, we describe the setup for our computational experiments and the benchmark methods used to evaluate the performance of \texttt{med.knn} in two case studies. In Section 4.4, we present the first case study on longitudinal data from the Framingham Heart Study (FHS). In Section 4.5, we present the second case study on electronic health records (EHR) data from the Dana Farber Cancer Institute. In Section 4.6, we discuss properties of our algorithm and key insights from our experiments. We conclude our work in Section 4.7.

4.2 Methods

In this section, we describe our proposed method for imputation. In Section 4.2.1, we define variables and notation that we use in this chapter. In Section 4.2.2, we introduce our framework for imputation MedImpute which directly models clinical covariates in time series, and we present the \(K\)-Nearest Neighbors (\(K\)-NN) based formulation. Finally, in Section 4.2.3, we provide the detailed steps of the first-order method \texttt{med.knn} that can be used to find high-quality solutions.

4.2.1 Variables and Notation

We introduce variables and define notation. We consider the single imputation problem for which our task is to fill in the missing values of data set \(X \in \mathbb{R}^{n \times p}\) with \(n\) observations (rows) and \(p\) features (columns). We assume that the first \(p_0\) features are continuous and that the next \(p_1 = p - p_0\) features are categorical. We suppose that the missing and known indices are specified by the following sets:

\[
\mathcal{M}_0 = \{(i, d) : \text{entry } x_{id} \text{ is missing, } 1 \leq d \leq p_0\},
\]

\[
\mathcal{N}_0 = \{(i, d) : \text{entry } x_{id} \text{ is known, } 1 \leq d \leq p_0\},
\]

\[
\mathcal{M}_1 = \{(i, d) : \text{entry } x_{id} \text{ is missing, } p_0 + 1 \leq d \leq p_0 + p_1\},
\]

\[
\mathcal{N}_1 = \{(i, d) : \text{entry } x_{id} \text{ is known, } p_0 + 1 \leq d \leq p_0 + p_1\}.
\]
In addition, we assume that all of the continuous variables are normalized with unit standard deviation and that the $d$th categorical variable takes value among $k_d$ classes. Given this data, we introduce the decision variables $W \in \mathbb{R}^{n \times p_0}$, $V \in \{1, \ldots, k_{p_0+1}\} \times \ldots \times \{1, \ldots, k_{p_0+p_1}\}$ to be the matrices of imputed continuous and categorical variables, respectively. For each entry $x_{id}$, $w_{id}$ is the imputed value if $d \in \{1, \ldots, p_0\}$, and $v_{id}$ is the imputed value if $d \in \{p_0 + 1, \ldots, p_0 + p_1\}$. Finally, let $\mathcal{I} = \{i : x_i \text{ has one or more missing values}\}$.

### 4.2.2 MedImpute

In this section, we present the MedImpute framework for imputation of clinical covariates in time series. We extend the general OptImpute framework by weighting instances of the same person in the imputation model. We focus on the $K$-NN classifier and provide the specific formulation to solve this problem. Our new framework takes into account the time series structure frequently encountered in healthcare data. In addition, unlike univariate time series methods, this approach leverages statistical correlations between multiple clinical covariates.

Suppose that we are given the same problem setup for single imputation as described in Chapter 3. In addition, assume that each observation $i$ corresponds to an individual $y_i \in \{1, \ldots, M\}$ recorded at a particular time point. For data sets with multiple observations of individuals over time, we have $M < n$. Define $t_i \in \mathbb{R}^+$ as the number of (days/months/years) after a reference date that observation $i$ was recorded. It follows that $|t_i - t_j|$ is the time difference in (days/months/years) between observations $i$ and $j$. Note that this framework captures the common structure of many clinical data sets collected over time, including longitudinal studies, insurance claims, and EHR data.

For each clinical covariate $d = 1, \ldots, p$, we introduce the parameters $\alpha_d, \lambda_d$. The first parameter $\alpha_d \in [0, 1]$ is the relative weight given to the time series component of the objective function for variable $d$. At the extremes, $\alpha_d = 0$ corresponds to imputing feature $d$ under the OptImpute objective, and $\alpha_d = 1$ corresponds to imputing feature $d$ using each individual’s time series information independently. The second parameter $\lambda_d \in (0, 1]$ is the exponential time decay parameter for variable $d$. We introduce this parameter so that observations from the same individual at nearby points in time will be weighted most heavily in the imputation.
For each pair of observations $i, j$, define

$$C_{ijd} = \begin{cases} 
\lambda_{d}|t_i - t_j|, & \text{if } y_i = y_j, \\
0, & \text{otherwise}.
\end{cases}$$

The constants $C_{ijd}$ will be coefficients in the time series component of the objective function. Therefore, in this model past observations of covariate $d$ from one week and one year ago would be given relative weights $\lambda_{d}^7$ and $\lambda_{d}^{365}$, respectively. We learn $\alpha_d$ and $\lambda_d$ via cross-validation.

The MedImpute formulation with the $K$-NN objective function is

$$\min \sum_{i \in I} \sum_{j=1}^{n} z_{ij} \left( \sum_{d=1}^{p_0} (1 - \alpha_d)(w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} (1 - \alpha_d)1_{\{v_{id} \neq v_{jd}\}} \right) + \sum_{i \in I} \sum_{j=1}^{n} \left( \sum_{d=1}^{p_0} \alpha_d C_{ijd}(w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} \alpha_d C_{ijd}1_{\{v_{id} \neq v_{jd}\}} \right)$$

s.t.

$$w_{id} = x_{id} \quad \forall (i, d) \in N_0,$$

$$v_{id} = x_{id} \quad \forall (i, d) \in N_1,$$

$$z_{ii} = 0 \quad \forall i \in I,$$

$$\sum_{j=1}^{n} z_{ij} = K \quad \forall i \in I,$$

$$Z \in \{0, 1\}^{|I| \times n},$$

where $I = \{i : x_i \text{ has one or more missing values}\}$ and $\alpha_d, C_{ijd}$ are constants. This problem is equivalent to (3.7) plus a penalty term in the objective for each feature $d$ with different weights $\alpha_d$ in order to account for instances of the same person in the data set. At the optimal solution, the objective function is the sum of the distances from each point to its $K$-nearest neighbors with respect to distance metric (3.6), plus the sum of the distances from each point to other observations from the same individual.

We derive a fast algorithm to provide high quality solutions to this problem using first order methods with random restarts, alternatively updating the binary variables and the imputed values as in opt.knn [22]. In Algorithm 2 we summarize the med.knn method for
a single warm start. In the next section, we describe the steps of this algorithm in detail.

MedImpute provides a flexible framework that can be easily extended as well. For example, we may consider other predictive models besides \( K \)-NN such as support vector machines and decision tree based methods by adjusting the objective functions of the corresponding OptImpute formulations appropriately. In these cases, we add the same penalty term to the objective functions that we added in formulation (4.1), and we solve using first-order methods with random starts. The method can also be adapted to a multiple imputation setting. However, while multiple imputation has been considered for several years to be the most accurate method for dealing with missing data [108], there is a tradeoff because single imputation is more interpretable. In particular, with single imputation we obtain one downstream predictive model that can be easily presented and explained to an entire clinical team, which is a critical step in the process of data-driven medical research [112].

Algorithm 2 med.knn

**Input:** Incomplete data matrix \( X \), threshold \( \delta_0 > 0 \), and warm start \([W^0, V^0]\).

**Output:** \( X^{imp} \) a full matrix with imputed values.

**Procedure:**

1. Initialize \( \delta \leftarrow \infty \), \( W^{old} \leftarrow W^0 \), \( V^{old} \leftarrow V^0 \).
2. **while** \( \delta > \delta_0 \) **do**
   1. Update \( Z^* \). For each \( i \in \mathcal{I} \), compute the distances \( \{d_{ij}\}_{j=1}^n \) between \((w_i^{old}, v_i^{old})\) and \((w_j^{old}, v_j^{old})\). Sort these values from lowest to highest, and set \( z_{ij}^* = 1 \) for the \( K \) smallest distances, and \( z_{ij}^* = 0 \) otherwise.
   2. Update the imputation \((W^*, V^*)\), following either BCD or CD. The details are provided in Section 4.2.3.
   3. \( \delta \leftarrow c(Z^{old}, W^{old}, V^{old}, X) - c(Z^*, W^*, V^*; X) \), where:

\[
\begin{align*}
c(Z, W, V; X) := & \sum_{i \in \mathcal{I}} \sum_{j=1}^n z_{ij} \left( \sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} 1_{\{v_{id} \neq v_{jd}\}} \right) \\
& + \sum_{i \in \mathcal{I}} \sum_{j=1}^n \left( \sum_{d=1}^{p_0} \alpha_d C_{ijd} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} \alpha_d C_{ijd} 1_{\{v_{id} \neq v_{jd}\}} \right) .
\end{align*}
\]

4. \( (Z^{old}, W^{old}, V^{old}) \leftarrow (Z^*, W^*, V^*) \).
5. **end while**
6. \( X^{imp} \leftarrow [W^*; V^*] \).
4.2.3 The med.knn algorithm

In this section, we provide details for the updates in the med.knn imputation algorithm. This is a first-order method to find locally optimal solutions to Problem (3.7). As in the opt.knn algorithm, in this algorithm we alternatively update $Z$ and $(W, V)$ until the solution converges. The update for $Z$ is identical to the one for opt.knn, and is computed with a simple sorting procedure on the distances. However, the update for $(W, V)$ is modified and depends upon the MedImpute parameters $\alpha_d$, $C_{ijd}$. As in opt.knn, we can update the values of $(W, V)$ either with Block Coordinate Descent (BCD) or Coordinate Descent (CD) which are described in the following subsections. The opt.knn updates for both BCD and CD are equivalent to the corresponding med.knn updates when $\alpha_d = 0$ for all $d = 1, \ldots, p$.

Block Coordinate Descent

In this approach, we update all of the imputed values at once. We call this approach BCD because we update the variables $(W, V)$ as an entire block, keeping $Z$ fixed. Our formulation Problem (4.1) decomposes by dimension into $p_0$ Quadratic Optimization problems for the continuous features and $p_1$ Mixed Integer Optimization problems for the categorical features. To update the imputed values $w^d$ for continuous feature $d = 1, \ldots, p_0$, we solve:

$$
\min_{w^d} \sum_{i \in I} \sum_{j=1}^n z_{ij}(1 - \alpha_d)(w^d_i - w^d_j)^2 + \sum_{i \in I} \sum_{j=1}^n \alpha_d C_{ijd}(w^d_i - w^d_j)^2 \\
\text{s.t.} \quad w^d_i = x^d_i \quad \forall (i, d) \in \mathcal{N}_0.
$$

(4.2)

Taking the partial derivative of the objective function with respect to $w^d_i$ for some missing entry $(i, d) \in \mathcal{M}_0$ and setting it to zero, we obtain after some simplifications:

$$
w^d_i = K + \sum_{j \in I} [(1 - \alpha_d)z_{ij} + \alpha_d C_{ijd}]w^d_i - \sum_{(j,d) \in \mathcal{M}_0} ((1 - \alpha_d)(z_{ij} + z_{ji}) + \alpha_d(C_{ijd} + C_{jdi})) \\
- \sum_{(j,d \in \mathcal{N}_0)} (\alpha_d C_{ijd} + (1 - \alpha_d)z_{ij} + 1_{\{j \in I\}}(1 - \alpha_d)z_{ji} + 1_{\{j \in I\}}\alpha_d C_{jdi}).
$$

(4.3)

For each feature $d = 1, \ldots, p_0$, we have a system of equations of the above form which we can
solve to determine the optimal imputed values \( w_{id}, (i, d) \in \mathcal{M}_0 \). Simplifying the notation, suppose that the missing values for the dimension \( d \) are \( \tilde{w}_d := (w_1d, \ldots, w_{ad}) \) and the known values are \( x^d := (x_{(a+1)d}, \ldots, x_{nd}) \). Then the set of optimal imputed values \( w^d_{id}, (i, d) \in \mathcal{M}_0 \) is the solution to the linear system

\[
((1 - \alpha_d)Q + \alpha_d P)\tilde{w}^d = ((1 - \alpha_d)R + \alpha_d Y)x^d,
\]

where the matrices \( Q, P, R, \) and \( Y \) are defined as

\[
Q = \begin{bmatrix}
    K + \sum_{j \in \mathcal{I}} z_{j1} - 2z_{11} & -z_{12} - z_{21} & \ldots & -z_{1a} - z_{a1} \\
    -z_{21} - z_{12} & K + \sum_{j \in \mathcal{I}} z_{j2} - 2(z_{22} + C_{22d}) & \ldots & -z_{2a} - z_{a2} \\
    \vdots & \vdots & \ddots & \vdots \\
    -z_{a1} - z_{1a} & -z_{a2} - z_{2a} & \ldots & K + \sum_{j \in \mathcal{I}} z_{ja} - 2z_{aa}
\end{bmatrix},
\]

\[
P = \begin{bmatrix}
    \sum_{j \in \mathcal{I}} C_{j1d} - 2C_{11d} & -C_{12d} - C_{21d} & \ldots & -C_{1ad} - C_{a1d} \\
    -C_{21d} - C_{12d} & \sum_{j \in \mathcal{I}} C_{j2d} - 2C_{22d} & \ldots & -C_{2ad} - C_{a2d} \\
    \vdots & \vdots & \ddots & \vdots \\
    -C_{a1d} - C_{1ad} & -C_{a2d} - C_{2ad} & \ldots & \sum_{j \in \mathcal{I}} C_{jad} - 2C_{aad}
\end{bmatrix},
\]

\[
R = \begin{bmatrix}
    z_{1(a+1)} + \mathbb{1}_{\{(a+1) \in \mathcal{I}\}} z_{(a+1)1} \ldots z_{1n} + \mathbb{1}_{\{n \in \mathcal{I}\}} z_{n1} \\
    \vdots & \ddots & \vdots \\
    z_{a(a+1)} + \mathbb{1}_{\{(a+1) \in \mathcal{I}\}} z_{(a+1)a} \ldots z_{an} + \mathbb{1}_{\{n \in \mathcal{I}\}} z_{na}
\end{bmatrix},
\]

\[
Y = \begin{bmatrix}
    C_{1(a+1)d} + \mathbb{1}_{\{(a+1) \in \mathcal{I}\}} C_{(a+1)1d} \ldots C_{1nd} + \mathbb{1}_{\{n \in \mathcal{I}\}} C_{n1d} \\
    \vdots & \ddots & \vdots \\
    C_{a(a+1)d} + \mathbb{1}_{\{(a+1) \in \mathcal{I}\}} C_{(a+1)ad} \ldots C_{and} + \mathbb{1}_{\{n \in \mathcal{I}\}} C_{nad}
\end{bmatrix}.
\]

Without loss of generality, there exists a closed-form solution

\[
\tilde{w}^d = ((1 - \alpha_d)Q + \alpha_d P)^{-1}((1 - \alpha_d)R + \alpha_d Y)x^d
\]

to this system of equations for each feature \( d = 1, \ldots, p_0 \). To update the imputed values
\(v^d\) for each categorical feature \(d = (p_0 + 1), \ldots, p\), we solve the following mixed-integer optimization problem:

\[
\begin{align*}
\min_{v^d} & \sum_{i \in \mathcal{I}} \sum_{j=1}^{n} ((1 - \alpha_d)z_{ij} + \alpha_dC_{ijd})y_{ij} \\
\text{s.t.} & \quad v_{id} = x_{id} \quad \forall (i, d) \in \mathcal{N}_1, \\
& \quad v_{id} - v_{jd} \leq y_{ij}k_d \quad \forall i, j, \\
& \quad v_{jd} - v_{id} \leq y_{ij}k_d \quad \forall i, j, \\
& \quad y_{ij} \in \{0, 1\}^{|\mathcal{I}| \times n}.
\end{align*}
\]

This is a Mixed Integer Optimization problem, which is practically solvable as the BCD update for \texttt{opt.knn}. Since the BCD update step requires inverting a matrix with \(O(n^2)\) entries and solving an optimization problem with \(O(n^2)\) binary variables, this method works best for smaller problem sizes \(n \leq 10,000\).

**Coordinate Descent**

In CD, we update the imputed values one at a time. In order to update the imputed value for \(x_{id}\), we fix all of the variables in Problem (4.1) except for \(w_{id}\) or \(v_{id}\) and solve the corresponding one-dimensional optimization problem. This results in fast, closed-form updates for both the continuous and categorical variables. Each \(w_{id}, (i, d) \in \mathcal{M}_0\) is imputed as the minimizer of the following:

\[
\begin{align*}
\min_{w_{id}} & \sum_{r \in \mathcal{I}} \sum_{j=1}^{n} z_{rj} \sum_{d=1}^{p_0} (1 - \alpha_d)(w_{rd} - w_{jd})^2 + \sum_{r \in \mathcal{I}} \sum_{j=1}^{n} \sum_{d=1}^{p_0} \alpha_dC_{rjd}(w_{rd} - w_{jd})^2. \\
\end{align*}
\] (4.4)

Solving the above gives the closed-form solution for every \((i, d) \in \mathcal{M}_0\):

\[
\begin{align*}
w_{id} = \frac{\sum_{j=1}^{n} ((1 - \alpha_d)z_{ij} + \alpha_dC_{ijd})w_{jd} + \sum_{j \in \mathcal{I}} ((1 - \alpha_d)z_{ji} + \alpha_dC_{jid})}{K + \sum_{j=1}^{n} \alpha_dC_{ijd} + \sum_{j \in \mathcal{I}} ((1 - \alpha_d)z_{ji} + \alpha_dC_{jid})}.
\end{align*}
\] (4.5)

Similarly, each categorical variable \(v_{id}, (i, d) \in \mathcal{M}_1\) is imputed as the minimizer of the fol-
\[
\min_{v_{id}} \sum_{r \in I} \sum_{j=1}^{n} z_{rj} \left\{ \begin{array}{c}
\sum_{d=p_0+1}^{p_0+p_1} (1 - \alpha_d) \mathbb{1}_{\{v_{rd} \neq v_{jd}\}} + \sum_{r \in I} \sum_{d=p_0+1}^{p_0+p_1} \alpha_d C_{rjd} \mathbb{1}_{\{v_{rd} \neq v_{jd}\}}.
\end{array} \right. 
\] (4.6)

Suppose that the value of categorical variable \(v_{id}\) is one of \(k_d\) distinct categories \(\{1, 2, \ldots, k_d\}\). Then, the solution to problem (4.6) is

\[
\arg \max_{k \in \{1, \ldots, k_d\}} \left[ \sum_{j=1}^{n} \left( (1 - \alpha_d) z_{ij} + \alpha_d C_{ijd} \right) \mathbb{1}_{\{v_{jd} = k\}} + \sum_{j \in I} \left( (1 - \alpha_d) z_{ji} + \alpha_d C_{ijd} \right) \mathbb{1}_{\{v_{jd} = k\}} \right]. 
\] (4.7)

Here, we set the imputed variable to be the value with the highest frequency in the neighborhood, with instances of the same person \(i\) receiving additional weight calibrated by the parameters \(C_{ijd})_{j=1}^{n}\) and \(\alpha_d\).

This approach scales to large problem sizes (\(n\) in the 100,000’s), and it is the method that we implement for the computational experiments.

### 4.3 Experimental Setup

In this section, we describe the setup of computational experiments that compare \texttt{med.knn} to state-of-the-art imputation methods. We use data from two distinct sources to test the performance of our algorithm on both longitudinal cohort study and electronic health records (EHR) data sets. The data sets along with the respective results are described in Sections 4.4 and 4.5.

In our experiments, we take the full data set to be the ground truth. We run some of the most commonly-used and state-of-the-art methods for imputation to predict the missing values and compare against \texttt{med.knn}. The methods that we compare are as follows:

1. **Mean** (\texttt{mean}): This is the simplest method. For each continuous feature, we impute the mean of the observed values and, for each categorical feature, we impute the mode of the observed values [91].
2. **Bayesian Principal Component Analysis** (bpca): This method takes a singular value decomposition (SVD) of the data matrix and information from a Bayesian prior distribution on the model parameters to impute missing values [99]. Implemented using the pcaMethods package in R.

3. **Multivariate Imputation via Chained Equations** (mice): In this multiple imputation method, we begin from \( m \) random starts and iteratively update each one to produce \( m \) independent imputations. In each iteration, we update the imputed values in feature \( d \) by drawing from a distribution conditional on all other features [125]. We use Classification Trees for the categorical features and Regression Trees for the continuous features. Implemented using the mice package in R.

4. **OptImpute under \( K \)-NN Objective** (opt.knn): This method finds a high quality solution to Problem (3.7) minimizing the sum of distances from each point to its \( K \)-Nearest Neighbors [22]. We find solutions to this problem using Algorithm 1 with the CD update. Fixing \( K = 10 \), we use several warm and random restarts and select the imputation with the best objective value. Implemented using the OptImpute package in Julia.

5. **MedImpute under \( K \)-NN Objective** (med.knn): This method finds a high quality solution to Problem (4.1) minimizing the sum of distances from each point to its \( K \)-Nearest Neighbors and other instances of the same individual. We find solutions to this problem using Algorithm 2 with the CD update. For each feature \( d \), we perform cross-validation to tune the parameters \( \alpha_d, \lambda_d \) with the rest of the MedImpute parameters set equal to zero. Fixing \( K = 10 \), we use several warm and random restarts and select the imputation with the best objective value. Implemented in Julia.

We evaluate the performance of all these methods on imputation accuracy, measuring the Mean Absolute Error (MAE). We also report their results on the corresponding downstream machine learning task, using the Area Under the Curve (AUC) metric. We compare the out-of-sample performance of an \( \ell_1 \)-regularized logistic regression model fit using the imputed data sets for various levels of missing information. We conduct 3 different types of
experiments that correspond to variations in the form of missing data that we encounter in medical data sets:

1. **Percentage of Missing Data**: We generate patterns of missing data for various percentages ranging from 10% to 50% under the missing completely at random (MCAR) mechanism. Given a target proportion of missing data \( f \) (i.e., \( f = 20\% \)), we generate among all observed data \( f \) missing values at each column independently from the rest completely at random.

2. **Number of Observations Per Patient**: With the missing percentage fixed at 50% MCAR, we vary the time frame during which patient observations are included in the imputation task. Our goal is to quantify the effect of the time series component as we vary its intensity.

3. **NMAR Mechanisms of Missing Data**: With the missing percentage fixed at 30%, we vary the missing data mechanism from missing completely at random (MCAR) to not missing at random (NMAR) on a gradient scale. In particular, we suppose that the missing pattern is \((\gamma 30\% \text{ NMAR}, (1 - \gamma)30\% \text{ MCAR})\), where \( \gamma \) varies from 0 to 1. We consider two different NMAR mechanisms that correspond to distinct missing data patterns observed in longitudinal studies and EHR. The two mechanisms are described in detail in subsections 4.4.2 and 4.5.2.

### 4.4 Longitudinal Data: Framingham Heart Study

In this section, we present a case study on longitudinal data from the Framingham Heart Study. In Section 4.4.1 we provide a description of the data set and the downstream tasks considered in the experiments. In Section 4.4.2 we give results where we compare the performance of \texttt{med.knn} to other state-of-the-art methods.
4.4.1 Data Description

The Framingham Heart Study (FHS) started in 1948 with the goal of observing a large population of healthy adults over time to better understand the factors that lead to cardiovascular diseases. Over 80 variables were collected from 5,209 people at a time for more than 40 years. The FHS is arguably the most influential longitudinal study in the field of cardiovascular and cerebrovascular research. This data has now been used in more than 2,400 studies and is considered one of the top 10 cardiology advances of the twentieth century alongside the electrocardiogram and open-heart surgery [78].

In this case study, we consider all individuals from the FHS Original Cohort [96] with 10 or more observations, which includes $M = 1,107$ unique patients. For each patient, we take the 10 most recent observations, so the data set has $n = 11,070$ observations total. We include $p = 13$ continuous (Age, Body Mass Index, Systolic Blood Pressure, High-Density Lipoproteins, Hematocrit, Blood Glucose levels) and categorical covariates (Gender, Smoking, presence of Cardiovascular Disease, presence of Atrial Fibrillation, presence of diabetes, currently under prescription of antihypertensive medication, presence of Left Ventricular Hypertrophy from ECG results).

We evaluate downstream models which predict 10-year risk of stroke given only the most recent visit for each participant. Therefore, while we use all 11,070 observations for the imputation task, we only use 1,107 observations for the downstream classification task. We consider this experimental setup so that the downstream classification task remains constant as the number of observations per patient (OPP) varies in the second set of experiments. In addition, these experiments demonstrate the marginal benefit of adding richer patient histories merely for the imputation task.

4.4.2 Results

In this section, we show the results from the computational experiments for the FHS. In Section 4.4.2, we provide results from experiments varying the percentage of missing data. In Section 4.4.2, we give results from experiments varying the number of observations per
patient. Finally, in Section 4.4.2 we give results from experiments varying the mechanism of missing data from MCAR to NMAR.

**Percentage of Missing Data**

We show the results from the FHS experiments with varying levels of missing data in Figure 4.1. Our method provides an edge over the rest of the methods in both imputation accuracy and performance on the downstream classification task. As the percentage of missing data increases, the relative improvement of \texttt{med.knn} over the reference methods increases. We report the absolute differences in MAE and AUC between \texttt{med.knn} and comparator methods in Table 4.1 along with the associated \( p \)-values.

![Image of Figure 4.1](image.png)

**Figure 4.1:** Results from experiments on the Framingham Heart Study data set with 10 observations per patient, varying the percentage of missing data from 10% to 50%.

In the FHS experiments with 50% missing data, the MAE of \texttt{med.knn} is 0.344, compared to 0.513 for the next best method \texttt{opt.impute}, a 32.9% reduction. Furthermore, the AUC of the logistic regression model trained using the \texttt{med.knn} imputation is 0.837, compared to 0.773 for the next best methods \texttt{mean} and \texttt{b pca}. These results demonstrate that \texttt{med.knn} is able to leverage time series information to gain a substantial edge over methods which ignore this information.
<table>
<thead>
<tr>
<th>Missing %</th>
<th>mice</th>
<th>bpcia</th>
<th>mean</th>
<th>opt.knn</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>-0.26 (&lt;0.001***)_</td>
<td>-0.24 (&lt;0.001***)_</td>
<td>-0.23 (&lt;0.001***)_</td>
<td>-0.18 (&lt;0.001***)_</td>
</tr>
<tr>
<td>20</td>
<td>-0.25 (&lt;0.001***)_</td>
<td>-0.23 (&lt;0.001***)_</td>
<td>-0.21 (&lt;0.001***)_</td>
<td>-0.18 (&lt;0.001***)_</td>
</tr>
<tr>
<td>30</td>
<td>-0.26 (&lt;0.001***)_</td>
<td>-0.22 (&lt;0.001***)_</td>
<td>-0.21 (&lt;0.001***)_</td>
<td>-0.18 (&lt;0.001***)_</td>
</tr>
<tr>
<td>40</td>
<td>-0.26 (&lt;0.001***)_</td>
<td>-0.21 (&lt;0.001***)_</td>
<td>-0.20 (&lt;0.001***)_</td>
<td>-0.18 (&lt;0.001***)_</td>
</tr>
<tr>
<td>50</td>
<td>-0.26 (&lt;0.001***)_</td>
<td>-0.19 (&lt;0.001***)_</td>
<td>-0.19 (&lt;0.001***)_</td>
<td>-0.17 (&lt;0.001***)_</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Missing %</th>
<th>mice</th>
<th>bpcia</th>
<th>mean</th>
<th>opt.knn</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0187 (&lt;0.001***)_</td>
<td>0.0109 (&lt;0.001***)_</td>
<td>0.0198 (&lt;0.001***)_</td>
<td>0.0386 (&lt;0.001***)_</td>
</tr>
<tr>
<td>20</td>
<td>0.0248 (&lt;0.001***)_</td>
<td>0.0220 (&lt;0.001***)_</td>
<td>0.0299 (&lt;0.001***)_</td>
<td>0.0425 (&lt;0.001***)_</td>
</tr>
<tr>
<td>30</td>
<td>0.0321 (&lt;0.001***)_</td>
<td>0.0328 (&lt;0.001***)_</td>
<td>0.0424 (&lt;0.001***)_</td>
<td>0.0542 (&lt;0.001***)_</td>
</tr>
<tr>
<td>40</td>
<td>0.0473 (&lt;0.001***)_</td>
<td>0.0437 (&lt;0.001***)_</td>
<td>0.0459 (&lt;0.001***)_</td>
<td>0.0720 (&lt;0.001***)_</td>
</tr>
<tr>
<td>50</td>
<td>0.0624 (&lt;0.001***)_</td>
<td>0.0641 (&lt;0.001***)_</td>
<td>0.0639 (&lt;0.001***)_</td>
<td>0.0888 (&lt;0.001***)_</td>
</tr>
</tbody>
</table>

Table 4.1: Pairwise t-tests between med.knn and benchmark methods for imputation and downstream classification tasks on the Framingham Heart Study data set, with the p-values adjusted for multiple comparisons.

**Number of Observations Per Patient**

In Figure 4-2, we show the results from the FHS experiments varying the number of observations per patient (OPP). When OPP = 1, med.knn is equivalent to the opt.knn method. We observe that the MAE of med.knn decreases significantly as OPP goes up to 4-5, and then increases slightly beyond this point. Because the observations in FHS data set occur every 2 years, this corresponds to data sets which include the past 6-8 years of clinical covariates for each individual. These results suggest that including observations earlier than this do not improve the med.knn imputation for this data set. Similarly, the AUC from med.knn peaks when OPP = 4 (6 years of past observations), and then levels off around 0.85. In contrast, the AUC of the benchmark methods declines slightly as OPP increases from 1-10, indicating that adding more observations to the data set does not help traditional imputation methods at any level.
Figure 4-2: Results from experiments on the Framingham Heart Study data set with 50% missing data, varying the number of observations per patient from 1 to 10.

NMAR Mechanism for Data from Longitudinal Studies

In longitudinal studies, missing data patterns often result from changes in the experiment design. Researchers may decide to include an additional set of variables as the study progresses over time due to new information from other investigations. Thus, it is common for feature \( d \) to be missing for the first \( t_d \) rounds of the longitudinal study. For example, ECG results were only first recorded in the FHS study 14 years after the study began [48, 88].

To generate \((\gamma30\% \text{ NMAR}, (1-\gamma)30\% \text{ MCAR})\) missing patterns under this mechanism, we apply the following process. We randomly select a covariate \( d \) and a discrete uniform random variable \( t_d \in \{1, 2, \ldots, 10\} \). The value \( t_d \) corresponds to the last round of the longitudinal study that covariate \( d \) is missing. For example, if \( t_d = 2 \) for the covariate “Left Ventricular Hypertrophy” (LVH), then the value for LVC will be missing for all observations in the two first clinical examinations. We continue this process until we have introduced \( \gamma30\% \text{ NMAR} \) missing values. Afterwards, we introduce \((1-\gamma)30\% \text{ MCAR} \) missing values to the remaining data set in order to obtain the final data set with 30% missing values.

In Figure 4-3, we show the results from the NMAR experiments on the FHS data set. We fix the total missing percentage to 30% and vary the \( \gamma \) parameter from 0 to 1. The performance of all methods remains relatively stable as we vary the mechanism of missing
data. Both in the imputation task and the downstream classification task, \texttt{med.knn} has a significant edge over the other methods for all values of $\gamma$. When $\gamma = 1$, the downstream AUC of \texttt{med.knn} is 0.879 compared to 0.847 for \texttt{mice}, a 3.2\% improvement. These experiments demonstrate that \texttt{med.knn} performs well given missing patterns with features missing sequentially in time.

![Figure 4-3: Results from experiments on the Framingham Heart Study data set with 30\% missing data, varying the ratio of each missing data mechanism from $\gamma = 0$ (30\% MCAR, 0\% NMAR) to $\gamma = 1$ (0\% MCAR, 30\% NMAR).](image)

4.5 Electronic Health Records: Dana Farber Cancer Institute

In this section, we present a case study on Electronic Health Records from the Dana Farber Cancer Institute. In Section 4.5.1, we provide a description of the data set and the downstream tasks considered in the experiments. In Section 4.5.2, we give results where we compare the performance of \texttt{med.knn} to other state-of-the-art methods.

4.5.1 Data Description

This data set was obtained from a recently published work on predicting mortality in late-stage cancer patients [13]. In this study, the authors retrospectively obtained patient data
from EHR and linked Social Security Administration mortality data for cancer patients at the Dana Farber / Brigham and Women’s Cancer Center from 2004 through 2014. Predictive models were fit for the entire population and individual cancers, including breast, lung, colorectal, kidney, and prostate cancer. Study eligibility required adult patients that have received at least one anticancer treatment over the course of their care, including chemotherapy, immunotherapy, and targeted therapy.

In this case study, we consider all patients with late-stage breast cancer in the Dana Farber EHR data set. Each observation corresponds to a patient initiating an anticancer regimen which was systematically recorded in the hospital’s database. As a result, for every patient who followed more than one regimen, multiple observations were collected. In total, we have 12,206 observations and 106 covariates that correspond to 5,987 unique patients. This includes 3,228 individuals who have just one line of therapy and therefore only appear once in this data set.

We evaluate downstream models which predict 60-day risk of mortality after the initiation of any anticancer regimen (not only the most recent), so we include multiple observations of the same individual in the training and testing sets. This matches how the predictive models were trained in the original work [13].

4.5.2 Results

In this section, we show the results from the computational experiments for the Dana Farber Cancer Institute. In Section 4.5.2, we provide results from experiments varying the percentage of missing data. In Section 4.5.2, we give results from experiments varying the number of observations per patient. Finally, in Section 4.5.2, we give results from experiments varying the mechanism of missing data from MCAR to NMAR.

Percentage of Missing Data

We show the results from the Dana Farber experiments with varying levels of missing data in Figure 4-4. For all missing percentages considered, med.knn produces the imputations with the highest accuracy and the best performance on the downstream classification task.
As the percentage of missing data increases, the relative improvement of med.knn over the reference methods increases. We report the absolute differences in MAE and AUC between med.knn and comparator methods in Table 4.2 along with the associated $p$-values.

![Figure 4-4: Results from experiments on the Dana Farber data set with entire EHR data for all patients, varying the percentage of missing data from 10% to 50.]

In the Dana Farber experiments with 50% missing data, the MAE of med.knn is 3.514, compared to 4.137 for the next best method mean. The AUC of the logistic regression model trained using the med.knn imputation is 0.901, compared to 0.896 for the next best method mice. These results demonstrate that med.knn gives a performance boost on a data set with heterogeneous numbers of observations per patient. The relative improvement of the algorithm is smaller here in comparison to the FHS case study because the majority patients have only one or just a few observations in this data set. As a result, there is less time series information available for the med.knn algorithm to exploit. In addition, the benchmark methods have higher out-of-sample AUC values, so an improvement of +0.5% in AUC is more significant for this downstream task.
Table 4.2: Pairwise t-tests between med.knn and benchmark methods for imputation tasks and downstream classification tasks on the Dana Farber EHR data set, with the p-values adjusted for multiple comparisons.

### Δ MAE (adjusted p-value)

<table>
<thead>
<tr>
<th>Missing %</th>
<th>mice</th>
<th>b pca</th>
<th>mean</th>
<th>opt.knn</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>-0.61 (0.018*)</td>
<td>-10.33 (&lt;0.001***)</td>
<td>-0.63 (0.018*)</td>
<td>-0.68 (0.015*)</td>
</tr>
<tr>
<td>20</td>
<td>-0.73 (&lt;0.001***)</td>
<td>-10.16 (&lt;0.001***)</td>
<td>-0.75 (&lt;0.001***)</td>
<td>-1.05 (0.001**)</td>
</tr>
<tr>
<td>30</td>
<td>-0.72 (&lt;0.001***)</td>
<td>-9.42 (&lt;0.001***)</td>
<td>-0.71 (&lt;0.001***)</td>
<td>-0.78 (&lt;0.001***)</td>
</tr>
<tr>
<td>40</td>
<td>-0.76 (&lt;0.001***)</td>
<td>-8.80 (&lt;0.001***)</td>
<td>-0.66 (&lt;0.001***)</td>
<td>-0.74 (&lt;0.001***)</td>
</tr>
<tr>
<td>50</td>
<td>-0.60 (&lt;0.001***)</td>
<td>-7.10 (&lt;0.001***)</td>
<td>-0.62 (&lt;0.001***)</td>
<td>-0.68 (&lt;0.001***)</td>
</tr>
</tbody>
</table>

### Δ AUC (adjusted p-value)

<table>
<thead>
<tr>
<th>Missing %</th>
<th>mice</th>
<th>b pca</th>
<th>mean</th>
<th>opt.knn</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0049 (&lt;0.001***)</td>
<td>0.0052 (&lt;0.001***)</td>
<td>0.0051 (&lt;0.001***)</td>
<td>0.0049 (&lt;0.001***)</td>
</tr>
<tr>
<td>20</td>
<td>0.0076 (&lt;0.001***)</td>
<td>0.0095 (&lt;0.001***)</td>
<td>0.0092 (&lt;0.001***)</td>
<td>0.0102 (&lt;0.001***)</td>
</tr>
<tr>
<td>30</td>
<td>0.0076 (&lt;0.001***)</td>
<td>0.0123 (&lt;0.001***)</td>
<td>0.0121 (&lt;0.001***)</td>
<td>0.0119 (&lt;0.001***)</td>
</tr>
<tr>
<td>40</td>
<td>0.0029 (0.073)</td>
<td>0.0133 (&lt;0.001***)</td>
<td>0.0131 (&lt;0.001***)</td>
<td>0.0134 (&lt;0.001***)</td>
</tr>
<tr>
<td>50</td>
<td>0.0059 (0.015*)</td>
<td>0.0174 (&lt;0.001***)</td>
<td>0.0167 (&lt;0.001***)</td>
<td>0.0182 (&lt;0.001***)</td>
</tr>
</tbody>
</table>

Number of Observations Per Patient

In Figure 4-5, we show results from the Dana Farber experiments varying the number of observations per patient (OPP) in the data set. Since in this data set the observations do not occur at regular intervals, we vary the time period of past EHR data considered for each patient instead of varying OPP directly. First, we notice that mice outperforms med.knn in the downstream classification task when the time period is less than 80 days. This is due to the fact that few patients have multiple observations within such a short time window. When the time period is 100 days or more, med.knn provides an edge of +0.5% in AUC.

In the imputation task, med.knn outperforms the benchmark methods for all time windows considered. We observe that med.knn consistently improves as the time period increases until it reaches 200 days, and then the MAE levels off. This may be due to the fact that the
proportion of patients who have recorded observations past this threshold is limited. Thus, the marginal benefit from those observations is limited since their number is insignificant.

Figure 4-5: Results from experiments on the Dana Farber data set with 50% missing data, varying the time period of past EHR data considered for each patient.

**NMAR Mechanism for Data from EHR**

In EHR data, missing patterns are usually correlated with the severity of patient’s condition. Consider the case of a patient whose physician suspects the existence of chronic kidney disease. The associated record is more likely to have a recorded value for Glomerular Filtration Rate since it is a direct indication of the kidney’s functional status [77]. Therefore, observed values are more likely to be below the threshold of \(60\) mL/min/1.73 m\(^2\) since they correspond to sicker patients.

To generate \(\gamma30\%\) NMAR patterns under this mechanism, we suppose that missing indicators are independent Bernoulli random variables where the probability that entry \(x_{id}\) is missing equals the probability that a normal random variable \(N(x_{id}, \epsilon)\) is greater than a particular threshold for covariate \(d\). The threshold for each covariate \(d\) is the quantile of \(X^d\) which corresponds to the desired missing percentage level \(\gamma30\%\). Then, we introduce \((1 - \gamma)30\%\) MCAR missing values to the remaining data set in order to obtain the final data set with 30% missing values total for this experiment.

In Figure 4-6, we show the results from the Dana Farber experiments varying the mech-
anism of missing data. We see that with the exception of bPCA, the MAE of all methods increases as $\gamma$ increases. In the downstream classification task, there is no discernible pattern and the AUC of all methods remains relatively constant. In particular, when $\gamma = 1$, the relative improvement in AUC of med.knn over the next best method mice is 1.0%, which is one of the largest improvements for this data set. This demonstrates that med.knn performs well under adversarial NMAR patterns such as this one because it is able to use the time series information effectively regardless of the missing data mechanism.

![Figure 4-6: Results from experiments on the Dana Farber data set with 30% missing data, varying the ratio of each missing data mechanism from $\gamma = 0$ (30% MCAR, 0% NMAR) to $\gamma = 1$ (0% MCAR, 30% NMAR).](image)

4.6 Discussion

Our computational experiments demonstrate that med.knn outperforms state-of-the-art methods imputing clinical covariates in time series. This method leverages the dependence between observations that correspond to the same individual over different points in time to obtain a more accurate single imputation. Further, downstream models trained on med.knn imputed data outperform competitor methods by a statistically significant amount. These results are consistent across a wide range of missing data patterns. MedImpute improves upon OptImpute by re-weighting the objective function to incorporate time series information. In this formulation, we introduce two parameters $\alpha_d, \lambda_d$ for each covariate $d$. We
are able to learn the values of these parameters via cross-validation for each covariate. In addition, these parameters are interpretable in a clinical context.

For example, we learn different values of $\alpha_d$ for chronic disease indicators such as Type 2 Diabetes Mellitus (T2DM) and lab values such as Systolic Blood Pressure (SBP). It is likely that an individual diagnosed with T2DM will continue to have this diagnosis regardless of the other covariates [1], so MedImpute finds $\alpha_d$ relatively close to 1 for this feature. On the other hand, the lab measurement of SBP may vary significantly during a single day [90], so previous observations of this covariate from the same individual provide relatively less information. For this feature, MedImpute finds $\alpha_d$ closer to 0 so that the $K$-nearest neighbors are weighted more heavily in the imputation. In addition, we learn $\lambda_d$ to determine the relative weights that we give to observations of feature $d$ from the same individual based on time elapsed. MedImpute selects higher values of $\lambda_d$ for features that change slowly over time such as the Body Mass Index and lower values for features that change rapidly over time such as SBP.

The relative improvement of med.knn over general purpose imputation methods increases as the amount of time series information increases. We see this effect comparing the results from the two case studies. In the FHS case study we consider participants who are recorded for 10 distinct visits occurring every 2 years. On the other hand, in the Dana Farber case study we consider patients who appear with variability in the data set. Each observation is associated with an anticancer regimen, which are strong treatments that can be very challenging for the human body. Therefore, the majority of individuals have only a single observation in the data set, and almost all have 5 or few observations. As a result, we observe that med.knn leads to smaller improvements in the imputation accuracy and downstream AUC for the Dana Farber case study. However, in a different EHR data set which includes more visits per patient, we expect that the benefit of our algorithm could be much greater.

In the NMAR experiments, we consider missing data mechanisms which are frequently encountered in practice. Longitudinal data sets often contain systematic missing information on some clinical examinations based on decisions made by the designers of the study. This is the case in the FHS where clinicians decided to steadily record an increasing number of variables that were suspected to be correlated with heart disease. As a result, forming a
retrospective data set results in values missing for all of the oldest observations. However, the missing data mechanisms of EHR data are typically quite different. In these data sets, the current condition of the patient has a significant impact upon the clinical covariates which are present and missing for each visit. Patients at higher risk are likely to undergo more detailed full medical examinations, providing richer records to their healthcare providers. In addition, clinicians may order lab tests for individuals who present with certain symptoms that are not be captured in the data. Through the NMAR experiments for each case study, we demonstrate that these imputation methods can work in real-world applications.

4.7 Conclusions

In this chapter, we propose the optimization framework MedImpute that addresses the missing data problem for multivariate data in time series encountered in medical applications. We introduce a new imputation algorithm \texttt{med.knn} that yields high quality solutions using optimization techniques combined with fast first-order methods. Through computational experiments with real-world data sets from the Framingham Heart Study and the Dana Farber Cancer Institute, we show that \texttt{med.knn} offers statistically significant gains in imputation quality over state-of-the-art imputation methods, which leads to improved out-of-sample performance on downstream tasks. The relative improvement grows as the percentage of missing data and the amount of time series information increases. As a flexible, accurate, and intuitive approach, MedImpute has the potential to become an indispensable tool for clinical applications with longitudinal missing data.
Chapter 5

Tensor Completion with Noisy Side Information

This work has been submitted for publication, with co-author Dimitris Bertsimas [21].

In this chapter, we develop a new model for tensor completion which incorporates noisy side information available on the rows and columns of a 3-dimensional tensor. This method learns a low rank representation of the data along with regression coefficients for the observed noisy features. Given this model, we propose an efficient alternating minimization algorithm to find high-quality solutions that scales to large data sets. We demonstrate that this method leads to significant gains in out-of-sample accuracy filling in missing values in both simulated and real-world data. In particular, we consider the problem of imputing drug response in two large-scale anti-cancer drug screens: the Genomics of Drug Sensitivity in Cancer (GDSC) and the Cancer Cell Line Encyclopedia (CCLE) data sets. On imputation tasks with 20% to 80% missing data, we show that the proposed method TensorGenomic outperforms all state-of-the-art methods including the original tensor model and a multilevel mixed effects model. With 80% missing data, TensorGenomic improves the $R^2$ from 0.404 to 0.552 in the GDSC data set and from 0.407 to 0.524 in the CCLE data set, compared to the tensor model which does not take into account genomic side information.
Mathematically, a tensor is a multidimensional array of numbers, typically with 3 or more dimensions\[^\text{73}\]. A vector is a 1-dimensional tensor, a matrix is a 2-dimensional tensor, and in general there are \(N\)-dimensional tensors. For example, suppose that we are given an e-commerce data set of \(n\) customers interacting with \(m\) products through \(\ell\) interactions. These interactions may include things such as: “searched for the product”, “purchased the product”, and “clicked on advertisement for the product”. We can represent this data as a 3-dimensional tensor \(Z \in \mathbb{R}^{n \times m \times \ell}\), where \(z_{ij}^k = 1\) if interaction \(k\) occurred for the pair (customer \(i\), item \(j\)) and \(z_{ij}^k = 0\) otherwise. This tensor may contain a large number of missing values, for instance because we have not shown advertisements to each (customer \(i\), item \(j\)) pair. This representation is useful because the data naturally varies along each dimension according to a different mechanism, which is the principal structure that is leveraged by mathematical models based on tensor data.

Given this tensor representation, we consider the problem of filling in the missing values of this tensor. This is known as the problem of tensor completion. In the e-commerce example, we would like to predict the purchase probability for each pair (customer \(i\), item \(j\)) so that we can display personalized advertisements and search recommendations. However, in order to develop the most accurate predictive model, in many cases it is insufficient to consider the tensor data in isolation because we have additional data available. Suppose that we are given additional data on the customers \(X \in \mathbb{R}^{n \times p}\) and additional data on the products \(Y \in \mathbb{R}^{m \times q}\) which are completely known. We refer to this additional data as side information. In practice, this side information may be noisy, which means that it contains only limited predictive power for the learning task at hand. Therefore, we will avoid making any strong assumptions about the relationships between \(X\), \(Y\), and \(Z\). In this work, we propose a model which leverages all of this data simultaneously to fill in the missing values of \(Z\).

As a real world application explored in this chapter, we consider the problem of personalized chemotherapy treatment for patients with cancer. Since data from human clinical trials is sparse in this area, we use data from large-scale anti-cancer drug screens, including the Genomics of Drug Sensitivity in Cancer (GDSC) and the Cancer Cell Line Encyclopedia (CCLE) data sets. These data sets are generated from in vitro experiments on cell lines,
which are samples of cells that have been taken from the tumors of patients with cancer and
grown in the lab [111]. Suppose that we are given a data set with \( n \) patients, \( m \) anti-cancer
drugs, and \( \ell \) doses. We can represent this data set as a tensor \( Z \in \mathbb{R}^{n \times m \times \ell} \), where \( z_{ijk} \) is
the percentage reduction in tumor size after the cell line from patient \( i \) receives anti-cancer
drug \( j \) at dose \( k \). In addition, we may also be given noisy side information in the form of
genomic features \( X \in \mathbb{R}^{n \times p} \) for the patients and drug target pathway features \( Y \in \mathbb{R}^{m \times q} \) for
the anti-cancer drugs. Our goal is to fill in the missing values in the tensor \( Z \) so that we
can prescribe the best anti-cancer treatment for each individual. While a lot of research has
been done on this subject, accurately predicting the response of an individual to anti-cancer
drugs remains a crucial challenge [2]. Furthermore, to our knowledge very little work has
been done trying to predict the response at particular doses. In computational experiments
in Section 5.3 we test tensor completion methods on the GDSC and CCLE data sets, and
we compare our approach to existing methods for this application.

5.0.1 Related Work

Our work belongs to the class of statistical methods known as collaborative filtering algo-
rithms [75, 74]. The objective of collaborative filtering is to learn the preferences of an
individual by collecting taste information from many individuals [39]. For instance, in the
previous two examples we were interested in learning the product preferences of consumers
and the drug preferences of cancer patients, respectively. Collaborative filtering methods
include algorithms for matrix completion and tensor completion, and typically use matrix
factorization methods [75].

There is extensive literature on the problem of matrix completion, with a surge in interest
starting in 2006 with the Netflix Prize competition [8]. In this competition, the internet
movie-streaming company Netflix asked participants to come up with a recommendation
system that accurately predicts movie ratings of users, with a $1 million dollar first prize.
The winning entry used a matrix factorization approach with modifications [4]. Over the past
decade, matrix completion methods have been used in many ratings-based recommendation
systems in e-commerce [136, 72].
Matrix factorization methods for matrix completion use the assumption that the underlying data is low rank. Intuitively, this means that the data matrix has a simpler structure than an arbitrary matrix with the same dimensions. Formally, the rank of a matrix $M \in \mathbb{R}^{n \times m}$ is the smallest integer $r$ such that it can be expressed as the product of two matrices $UV^T$, where $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{m \times r}$. Computationally efficient methods are available for learning low rank matrix approximations, including nuclear-norm minimization, singular-value decomposition, and alternating minimization. These approaches are widely used for solving the problem of matrix completion \cite{38, 37, 34, 89, 67}.

In addition, matrix completion methods that incorporate side information have been studied. For example, Inductive Matrix Completion (IMC) is a method for matrix completion with exact side information \cite{66}. In this model, we assume that the data matrix $M \in \mathbb{R}^{n \times m}$ can be expressed as the product of $XSY^T$, where $X \in \mathbb{R}^{n \times p}$, $Y \in \mathbb{R}^{m \times q}$ are the matrices of side information and $S \in \mathbb{R}^{p \times q}$ is learned from the data. Alternatively, \cite{45} proposed a method given noisy side information which uses weaker assumptions on the data structure. In their model, they assume that the data matrix can be expressed as $XSY^T + R$, where $X$, $S$, $Y$ are defined as before and $R \in \mathbb{R}^{n \times m}$ is a low rank component learned from the data. In our approach, we use a similar additive model to incorporate noisy side information, but extended to the tensor setting. Finally, we note that there are several other methods, including Kernelized Bayesian Matrix Factorization which integrates side information using Bayesian priors \cite{59}, and extensions of IMC which impose sparsity constraints upon the $S$ matrix \cite{86, 18}.

In order to extend the ideas from matrix completion to tensor completion, the first thing required is a generalization of the concept of matrix rank to higher dimensions. There are multiple definitions for the rank of a tensor with 3 or more dimensions, including the CP rank, Tucker rank, and Slice rank \cite{73, 123, 51}. Some of these objects such as the Tucker rank have multiple components based upon the number of dimensions in the tensor. Similar to the matrix rank, if a tensor $Z \in \mathbb{R}^{n \times m \times \ell}$ has low rank according to one of these criteria, then it has a simpler structure than an arbitrary tensor with the same dimensions. We discuss the mathematical properties of tensors in more detail in Section 5.1.1 and we provide the formal definitions of these concepts of tensor rank in Appendix B.1.
Previous work has been done to find the best low rank approximation to a tensor for different definitions of the tensor rank. The CP and Tucker decompositions are well-known \cite{73}. However, these methods are computationally intensive and impractical for large-scale data sets. Several promising results in recent years have focused on finding the best convex tensor approximation by minimizing the sum or a convex combination of the components of the Tucker rank \cite{57,84}. These algorithms have recovery guarantees and generalize better than exact Tucker decomposition when the number of observed entries is greater than a certain threshold \cite{120}.

Newer approaches which use non-convex approaches have been shown to outperform convex methods for tensor completion. \cite{44} propose a non-convex projected gradient descent method which bounds the Tucker rank and imposes sparsity on the tensor approximation. In addition, \cite{51} propose a non-convex method for 3-dimensional tensor completion which provides stronger statistical guarantees compared to general methods for $n$-dimensional tensors. Their proposed algorithm learns a low Slice rank representation of the data via a hard-thresholding SVD approach which can scale to large data sets. In this chapter, we also restrict our focus to the 3-dimensional tensor completion problem, and we extend the model proposed by \cite{51} to accommodate noisy side information on the rows and the columns of the tensor.

There has also been some previous work adapting tensor completion methods to accommodate side information. For example, \cite{93} propose a method that finds the best CP or Tucker approximation with an additional regularization term based on graph Laplacians to incorporate the side information. This method is slightly less computationally efficient compared to the original CP and Tucker decomposition algorithms. \cite{107} propose a completely Bayesian model that enriches the original CP decomposition with a second-layer tensor decomposition that incorporates side information. However, this method requires many tuning parameters, and in practice we may not have the distributional information which is required for the Bayesian priors. In general, current methods for tensor completion which account for side information are computationally intensive and difficult to implement for problems encountered in practice.
Finally, we outline the literature which is related to the real-world application that we consider. Many collaborative filtering methods have been applied to predict gene-disease associations and drug response. For example, IMC and probability-based collaborative filtering have been used to discover gene-disease associations in the Online Mendelian Inheritance in Man (OMIM) database [61, 94, 137]. In addition, several methods have been developed specifically to predict anti-cancer drug response in the GDSC and CCLE data sets. For example, [116] extend Kernelized Bayesian Matrix Factorization to predict anti-cancer drug response in the GDSC data set leveraging drug pathway data as side information. [83] propose a nearest-neighbors based method which incorporates genomic and drug side information into a low rank model. Other methods which do not rely upon low rank models have also been developed to predict anti-cancer drug response, including random forest, deep learning, and network-based methods [106, 115, 54].

Our approach for predicting anti-cancer drug response differs from the above methods because we train on the raw experimental data from the drug screens, which is the drug response of cell lines from patients with solid tumor cancers to anti-cancer treatments at particular doses. As a result, the training data is a 3-dimensional tensor with dimensions (patient, drug, dose). In contrast, the previous methods rely upon a pre-processing step to determine the sensitivity of each (patient, drug) pair first. These sensitivity values are taken as ground truth, and then matrix completion methods are fit on top. In this work, we avoid the dependence upon intermediate models by casting this as a tensor completion problem instead of a matrix completion problem.

5.0.2 Contributions

The contributions of this chapter are as follows:

1. We propose an extension to the low rank model for tensor completion proposed by [51] that leverages noisy side information. In particular, we propose a model for tensor completion with one-sided information that incorporates noisy features of the rows, and a model for tensor completion with two-sided information that incorporates noisy
features of both the rows and columns. Each model is composed of a low rank component which leverages the structure of the observed values in the tensor and a regression component which leverages the noisy side information.

2. For each model, we derive fast algorithms based upon alternating minimization which find high quality solutions. In particular, we present the algorithms TensorOneSided and TensorTwoSided for tensor completion given noisy one-sided and two-sided information, respectively.

3. In experiments on simulated data, we demonstrate that the proposed method TensorTwoSided significantly outperforms benchmark methods for tensor completion given two-sided information with varying levels of noise. The benchmark methods considered include the original tensor completion method Tensor which does not incorporate side information and a regression method TwoSided which uses side information only.

4. In experiments on real-world data, we demonstrate that the proposed method TensorGenomic outperforms state-of-the-art methods for predicting anti-cancer drug response in the Genomics of Drug Sensitivity in Cancer (GDSC) and Cancer Cell Line Encyclopedia (CCLE) data sets with 20% to 80% missing values given genomic side information. In particular, with 80% missing data, TensorGenomic improves the $R^2$ from 0.404 to 0.552 in the GDSC data set and improves the $R^2$ from 0.407 to 0.524 in the CCLE data set compared to the low rank tensor model which does not take into account genomic side information.

The structure of this chapter is as follows. In Section 5.1 we describe our proposed methods for tensor completion for problems with noisy one-sided and two-sided information. In Section 5.2 we compare the performance of our methods against benchmark tensor completion methods on simulated data experiments. In Section 5.3 we test the performance of the method for tensor completion on two real-world examples predicting anti-cancer drug response with genomic side information. In Section 5.4 we discuss the results from the simulated and real-world computational experiments. We conclude in Section 5.5.
5.1 Methods

In Section 5.1.1, we provide some background material on tensors which is prerequisite material for this work. In Section 5.1.2, we state the problem of tensor completion with noisy side information. In Sections 5.1.3 and 5.1.4, we introduce two basic regression models for tensor completion using one-sided and two-sided information, and we present two fast methods based upon accelerated gradient descent. In Section 5.1.5, we introduce a low rank model for tensor completion without side information, and we review the Slice Learning method. In Section 5.1.6, we introduce a low rank model for tensor completion with one-sided information that uses features on the rows, and we present the method $\text{TensorOneSided}$. In Section 5.1.7, we introduce a low rank model for tensor completion with two-sided information that uses features on both rows and columns, and we present the method $\text{TensorTwoSided}$.

5.1.1 Background on Tensors

In this section, we cover a few preliminaries on tensors and the notation that we use to describe them. A tensor is a multidimensional array or $N$-way array $[73]$. In this work, we consider only 3-way tensors in the Euclidean space $\mathbb{R}^{n \times m \times \ell}$. We refer to $n$, $m$, and $\ell$ as the number of rows, columns, and slices of the tensor, respectively. For a given tensor $Z \in \mathbb{R}^{n \times m \times \ell}$, let $z_{ij}^k$ be the element in the $i$th row, $j$th column, and $k$th slice of the tensor. In addition, we refer to the matrix formed by the $k$th slice of the tensor as $Z^k \in \mathbb{R}^{n \times m}$. If $Z$ has missing values, we denote the known and missing entries in the $k$th slice of the tensor as

$$\Omega_k = \{(i, j) : z_{ij}^k \text{ is known}\},$$
$$\Omega_k^c = \{(i, j) : z_{ij}^k \text{ is missing}\}.$$

and across all slices of the tensor as

$$\Omega = \{(i, j, k) : z_{ij}^k \text{ is known}\},$$
$$\Omega^c = \{(i, j, k) : z_{ij}^k \text{ is missing}\}.$$

We also describe some basic notation that we use to refer to matrices. For a matrix
Figure 5-1: Visualizations of a 3-dimensional tensor and its mode-1 and mode-2 unfoldings.

\[ \mathbf{X} \in \mathbb{R}^{n \times p}, \text{ let } x_{id} \text{ be the element in the } i \text{th row and } d \text{th column. We denote the transpose of } \mathbf{X} \text{ as } \mathbf{X}^T \in \mathbb{R}^{p \times n} \text{ and the column rank as } \text{rank}(\mathbf{X}). \text{ We also will use the Frobenius norm of a matrix, which is defined as} \]

\[ \| \mathbf{X} \|_F := \sqrt{\sum_{i,d} x_{id}^2}. \]

Tensor Unfoldings

Next, we define the *unfoldings* of a tensor \( \mathbf{Z} \in \mathbb{R}^{n \times m \times \ell} \). We define the mode-1 unfolding of a tensor to be the horizontal concatenation of the slices of the tensor into a single matrix. We denote this matrix as \( \mathbf{Z}_{(1)} \in \mathbb{R}^{n \times m\ell} \). In \( \mathbf{Z}_{(1)} \), the columns are equivalent to the columns of all of the slices in the original tensor. Likewise, we define the mode-2 unfolding of a tensor to be the horizontal concatenation of the transposed slices into a single matrix. We denote the mode-2 unfolding as \( \mathbf{Z}_{(2)} \in \mathbb{R}^{m \times n\ell} \). In \( \mathbf{Z}_{(2)} \), the columns are equivalent to the rows of all of the slices in the original tensor. Following this pattern, we can also define the mode-3 unfolding of a tensor, although forming this object requires breaking up the tensor slices. We consider the vector \( \mathbf{z}_{ij} \) formed by fixing the \( i \)th row and \( j \)th column, and then varying the third dimension. The mode-3 unfolding \( \mathbf{Z}_{(3)} \in \mathbb{R}^{\ell \times nm} \) is the matrix formed by horizontally concatenating all of these vectors \( \mathbf{z}_{ij} \). In Figure 5-1 we provide visualizations of a 3-dimensional tensor and its mode-1 and mode-2 unfoldings. For more discussion on tensor unfoldings, we refer the reader to [73]. In Appendix B.1 we provide definitions for the rank of a tensor which use these concepts of tensor unfoldings.
5.1.2 Tensor Completion Problem

In this section, we state the problem of tensor completion with noisy side information.

Suppose that we are given a 3-dimensional data set $Z \in \mathbb{R}^{n \times m \times \ell}$ with known and missing values $\Omega_k, \Omega_k^c$ for each slice $k$, respectively. In addition, suppose that we are also given noisy features $X \in \mathbb{R}^{n \times p}$ and $Y \in \mathbb{R}^{m \times q}$ of the rows and columns of this data, respectively.

For example, consider an e-commerce data set with $n$ customers, $m$ products, and $\ell$ interactions. In this tensor $Z$, $z_{ij}^k = 1$ if customer $i$ interacted with product $j$ via interaction $k$, and $z_{ij}^k = 0$ otherwise. Many of the entries of $Z$ are missing because each customer typically has only a few interactions with a subset of the products. In addition, we have $p$ features of the customers, such as the age, gender, location, and electronic device of each customer, which are captured in the row side information $X$. We also have $q$ features of the products, such as the brand, supplier, and average review score of each product, which are captured in the column side information $Y$.

As another example, consider a drug screening data set with $n$ patients, $m$ drugs, and $\ell$ doses. In this tensor, $z_{ij}^k$ is the outcome when patient $i$ is treated with drug $j$ at dose $k$. Many of the entries of $Z$ are missing because each patient has only received a few (drug, dose) treatment combinations in the past. In the row side information $X$, we have $p$ features of the patients, which may include demographic or genomic data. In the column side information $Y$, we have $q$ features of the drugs, which may include drug target pathway data.

Given the observed values of $Z$ and the noisy features $X$, $Y$, our goal is to find the approximation $\hat{Z} \in \mathbb{R}^{n \times m \times \ell}$ which is as close as possible to the original $Z$. In particular, we would like to find $\hat{Z}$ which minimizes the sum-of-squared errors across all of the slices:

$$\sum_{k=1}^{\ell} \|Z^k - \hat{Z}^k\|_F^2. \quad (5.1)$$
5.1.3 One-Sided Regression Model

In this section, we introduce the one-sided regression model which uses row side information only, and we present the OneSided method. This is a simple model based upon linear regression which does not leverage information across multiple slices of the tensor.

Suppose that \( \mathbf{x}_i \in \mathbb{R}^p \) is the vector of features for the \( i \)th row in the tensor. Consider the linear model:

\[
\hat{z}_{ij}^k = \mathbf{x}_i^T \mathbf{w}_{ij}^k,
\]

where \( \mathbf{w}_{ij}^k \in \mathbb{R}^p \) are weights for a particular (column, slice) pair. We can interpret the weight \( w_{ij}^k \) as the amount of change in the prediction given one unit increase in \( x_{id} \). In tensor
notation, we have

\[ \hat{Z}^k = XW^k, \]  

(5.3)

where \( W^k \in \mathbb{R}^{p \times m} \) is the matrix of weights for the \( k \)th slice. We can learn \( W \) by running \( m\ell \) independent linear regressions, one for each (column, slice) pair. We can fit all of these linear regression models simultaneously by considering the following optimization problem:

\[
\min \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} (z_{ij}^k - (XW^k)_{ij})^2 + \frac{1}{\gamma} \|W^k\|_F^2,
\]  

(5.4)

where \( \gamma \) is a regularization parameter. Problem (5.4) is a quadratic optimization problem which is efficiently solvable. In particular, we solve this subproblem using Nesterov’s accelerated gradient descent method [97]. Let \( f(W; X, \Omega) \) be the objective function of problem (5.4). The partial derivative of \( f \) with respect to \( w_{dj}^k \) is

\[
\frac{\partial f(W; X, \Omega)}{\partial w_{dj}^k} = \frac{2}{\gamma} w_{dj}^k + \sum_{i:(i,j) \in \Omega_k} 2x_i^T(x_i^T W_{ij}^k - z_{ij}^k).
\]

Let \( \nabla f(W; X, \Omega) \) be the full gradient of \( f \) with respect to \( W \). In Algorithm 3, we present an accelerated gradient descent method for solving problem (5.4) using this gradient. We can further speed up this method by selecting the step size \( \nu \) dynamically at each step via backtracking line search [98].

5.1.4 Two-Sided Regression Model

In this section, we introduce the two-sided regression model which uses both row and column side information, and we present the TwoSided method. Similar to the one-sided model, this model does not leverage information across multiple slices of the tensor.

Suppose that \( x_i \in \mathbb{R}^p \) is the vector of features for the \( i \)th row, and \( y_j \in \mathbb{R}^q \) is the vector of features for the \( j \)th column. Consider the bilinear model:

\[ z_{ij}^k = x_i^T W^k y_j, \]  

\[ 170 \]
Algorithm 3 OneSided

**Data:** Tensor $Z \in \mathbb{R}^{n \times m \times \ell}$ with known entries

$\Omega = \{(i, j, k) : z_{ij}^k \text{ is known}\}$,

side information $X \in \mathbb{R}^{n \times p}$.

**Input:** Warm start $W_0 \in \mathbb{R}^{p \times m}$,

regularization parameter $\gamma > 0$,

max number of gradient steps $G \geq 1$, step size $\nu > 0$.

**Output:** Optimal solution to problem (5.4).

**Procedure:**

Initialize $W_1 \leftarrow W_0$, $t \leftarrow 1$.

while $t < G + 1$ do

$\hat{W} \leftarrow W_t + \frac{t-1}{t+2} (W_t - W_{t-1})$,

$W_{t+1} \leftarrow \hat{W} - \nu \nabla f(W; X, \Omega)$,

$t \leftarrow t + 1$.

end while

return Estimated value of $W$.

---

where $W^k \in \mathbb{R}^{p \times q}$ are weights for the $k$th slice of the tensor. We can interpret the weight $w^k_{de}$ as the amount of change in the prediction given one unit increase in the interaction term $x_{id}y_{je}$. In tensor notation we have

$$\hat{Z}^k = XW^kY$$

for each slice $k = 1, \ldots, \ell$. In order to find the weights $W$, we consider the following optimization problem:

$$\min_{W} \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} (z_{ij}^k - (XW^kY)_{ij})^2 + \frac{1}{\gamma} \|W^k\|_F^2,$$  \hspace{1cm} (5.5)

where $\gamma$ is a regularization parameter. This is a quadratic optimization problem nearly identical to the one-sided regression formulation. If $Y$ is the $m \times m$ identity matrix, then these two formulations are equivalent. Let $g(W; X, Y, \Omega)$ be the objective function of problem (5.5), and let $\nabla g(W; X, Y, \Omega)$ be the gradient of $g$ with respect to $W$. In Algorithm 4, we present an accelerated gradient descent method for solving problem (5.5) using this gradient.
Algorithm 4 TwoSided

**Data:** Tensor $Z \in \mathbb{R}^{n \times m \times \ell}$ with known entries
\[ \Omega = \{(i, j, k) : z_{ij}^k \text{ is known}\}, \]
side information $X \in \mathbb{R}^{n \times p}$, $Y \in \mathbb{R}^{m \times q}$.

**Input:** Warm start $W_0 \in \mathbb{R}^{p \times q}$,
regularization parameter $\gamma > 0$,
max number of gradient steps $G \geq 1$, step size $\nu > 0$.

**Output:** Optimal solution to problem (5.5).

**Procedure:**
- Initialize $W_1 \leftarrow W_0$, $t \leftarrow 1$.
- while $t < G + 1$ do
  - $W \leftarrow W_t + \frac{t}{t+2}(W_t - W_{t-1})$,
  - $W_{t+1} \leftarrow W - \nu \nabla g(W; X, Y, \Omega)$,
  - $t \leftarrow t + 1$.
- end while
- return Estimated value of $W$.

5.1.5 Basic Tensor Model

In this section, we introduce a low rank model for tensor completion without side information, and we present the Tensor method. This low rank model is equivalent to the one originally proposed by [51].

There are two shortcomings of the one-sided and two-sided regression models that we have presented so far. First of all, these models do not leverage information across multiple slices of the tensor to impute the missing values. Second, because the observed row and column features are noisy, they are typically poor predictors for the tensor on their own. For example, in a drug screening data set, genomic features are typically poor predictors of drug response on their own. The Tensor model that we present next addresses these issues.

Instead of trying to improve the noisy row and column features, we will try to learn new features from scratch using only the observed values in the tensor. In the Tensor model, we suppose that there are a few true underlying latent features of the rows and columns which are constant across all slices of the tensor. We assume that there are at most $r$ latent features for the rows and at most $r$ latent features for the columns, and all of these latent features are unknown a priori. This is known as a low rank assumption, which is commonly used for collaborative filtering methods [75, 74].
Let $u_i \in \mathbb{R}^r$ be the latent features of row $i$ and let $v_j \in \mathbb{R}^r$ be the latent features of observation $j$. Given these latent features, the generative model for $z^k_{ij}$ is

$$
\hat{z}_{ij}^k = u_i^T S^k v_j,
$$

(5.6)

where $S^k \in \mathbb{R}^{r \times r}$ is a matrix of fitted coefficients. Let $U \in \mathbb{R}^{n \times r}$ be the matrix of row latent features and let $V \in \mathbb{R}^{m \times r}$ be the matrix of column latent features. It follows that the model for the $k$th slice of the tensor is

$$
\hat{Z}^k = U S^k V^T.
$$

(5.7)

For different slices, the latent features $U$ and $V$ remain the same, but the fitted coefficients $S^k$ are different. This structural assumption is equivalent to requiring that the Slice rank of $\hat{Z}$ is at most $r$. In Figure 5-2, we show a diagram of this tensor model for the drug screening data set.

To find $U$, $S$, $V$, we consider the following optimization problem:

$$
\min_{U,S,V} \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} (z^k_{ij} - (US^k V^T)_{ij})^2.
$$

(5.8)

Note that this formulation requires a single parameter, the tensor rank $r$, which we will learn via cross-validation.
Unlike the previous one-sided and two-sided regression formulations, problem (5.8) is nonconvex, so we cannot compute the global optimal solution. However, we can find high-quality solutions via nonconvex methods. In particular, we can use an iterative procedure based upon the Slice Learning algorithm proposed by [51] to find high-quality solutions. In this procedure, we begin with a warm start solution $\hat{Z}$. Each iteration, we run the Slice Learning algorithm and update $\hat{Z}$ in the missing entries $\Omega^c$ of the original tensor. We repeat until the tensor approximation $\hat{Z}$ converges to a stationary point.

In a single iteration, we run the Slice Learning algorithm to obtain updated estimates for $U$, $V$, and $S^1, \ldots, S^\ell$. First, we estimate the latent features of the rows $U$ by taking the singular value decomposition (SVD) of the mode-1 unfolding of $\hat{Z}$. Let $U_1 \Sigma_1 V_1^T$ be the SVD of $\hat{Z}_{(1)}$, where $U_1$, $V_1$ are orthonormal and $\Sigma_1$ is diagonal. We set $U$ to be the $r$ columns of $U_1$ which correspond to the top $r$ singular values. We denote this operation as:

$$U \leftarrow \text{svds}(\hat{Z}_{(1)}, r).$$

Similarly, we estimate the latent features of the columns $V$ by taking the SVD of the mode-2 unfolding of $\hat{Z}$. The update for $V$ is

$$V \leftarrow \text{svds}(\hat{Z}_{(2)}, r).$$

Finally, we update the estimates for $S^1, \ldots, S^\ell$. Since $U$, $V$ are orthonormal, we have $U^{-1} = U^T$ and $V^{-1} = V^T$. Therefore the update for $S^k$ which minimizes the squared error for slice $k$ is

$$S^k \leftarrow U^T \hat{Z}^k V.$$

In Algorithm 5, we summarize this method for tensor completion without side information. In the next two sections, we see how this method can be modified to incorporate side information on the rows and/or columns.
Algorithm 5 Tensor Model

**Data:** Tensor $Z \in \mathbb{R}^{n \times m \times \ell}$ with missing entries

$\Omega^c = \{(i, j, k) : z_{ijk}^c \text{ is missing}\}$.

**Input:** Rank $r$, warm start $Z_0 \in \mathbb{R}^{n \times m \times \ell}$, max number of iterations $T \geq 1$.

**Output:** Locally optimal solution to problem (5.8).

**Procedure:**

Initialize $\hat{Z} \leftarrow Z_0$, $t \leftarrow 0$.

while $t < T$ do

$R \leftarrow \hat{Z}$,

$U \leftarrow \text{svds}(R(1), r)$,

$V \leftarrow \text{svds}(R(2), r)$,

$S^k \leftarrow U^T R^k V$ \quad $\forall k$,

$\hat{z}_{ijk}^k \leftarrow (U S^k V^T)_{ij}$ \quad $\forall (i, j, k) \in \Omega^c$,

$t \leftarrow t + 1$.

end while

return Estimated values of $U, S, V$.

5.1.6 Tensor Model with Noisy One-Sided Information

In this section, we introduce a low rank model for tensor completion given noisy one-sided information, and we present the method **TensorOneSided**. This model combines components from the **Tensor** and **OneSided** models.

In this approach, we model the tensor as the sum of two components, with one component that we learn from the Slice learning decomposition and one component that we learn from the row side information. Let $x_i$ be the observed features of row $i$. The resulting generative model for $z_{ijk}^k$ is

$$
\hat{z}_{ijk}^k = u_i^T S^k v_j + x_i^T w_{jk}^k
$$

(5.9)

where $u_i$ are latent features of row $i$, $v_j$ are latent features of row $j$, $S^k$ are weights of the latent features for slice $k$, and $w_{jk}^k$ are (column, slice)-specific weights. It follows that the model for the $k$th slice of the tensor is

$$
\hat{Z}^k = U S^k V^T + X W^k,
$$

(5.10)

where $U \in \mathbb{R}^{n \times r}$, $V \in \mathbb{R}^{m \times r}$, $S^1, \ldots, S^\ell \in \mathbb{R}^{r \times r}$, and $W^1, \ldots, W^\ell \in \mathbb{R}^{p \times m}$ are learned.
Figure 5-3: Tensor model of a drug screening data set with noisy side information for the patients only. \( n \) is the number of patients, \( m \) is the number of drugs, \( \ell \) is the number of doses tested, \( r \) is the number of latent features, and \( p \) is the number of observed patient features.

from the data. We can interpret model (5.10) as the basic tensor model (5.7) with an additional term to predict the residuals that is linear with respect to the observed row features. Note that if \( W = 0 \), then this model reduces to the the basic tensor model exactly. This is important because in some cases the side information may not provide any additional predictive power. Similarly, if any of \( U, S, V \) are equal to zero, then this reduces to the regression model (5.4) using row side information only. In Figure 5-3, we show a diagram of this tensor model with one-sided information for a drug screening data set.

To find \( W, U, S, V \), we consider the following optimization problem:

\[
\min_{W,U,S,V} \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} \left( z_{ij}^k - (US^kV^T + XW^k)_{ij} \right)^2 + \frac{1}{\gamma} \|W^k\|_F^2, \tag{5.11}
\]

where \( \gamma \) is a regularization parameter. This formulation uses two parameters \( \gamma \) and \( r \) which we can learn via cross-validation. Taking the limit as \( \gamma \to 0 \), this model reduces to the original tensor formulation (5.8).

We propose the following alternating optimization procedure to solve problem (5.11), which we refer to as **TensorOneSided**. In this approach, we alternate between running the Slice Learning algorithm and solving a quadratic optimization problem.

1. Begin with a warm start solution \( \hat{Z} \). Initialize all of the variables \( W, U, S, V \) to zero.
2. Update $U, S, V$ by considering the following problem:

$$\min_{U, S, V} \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} \left( (\hat{Z}^k - X W^k)_{ij} - (US^k V^T)_{ij} \right)^2.$$  \hspace{1cm} (5.12)

We can find high-quality solutions to this problem using the Slice Learning algorithm [51]. Let $R$ be the tensor of residuals, where $R^k = \hat{Z}^k - X W^k$. In this step, we find a low rank tensor approximation to $R$ by taking SVDs of the mode-1 and mode-2 unfoldings.

3. Update the $W$ by considering the following problem:

$$\min_{W} \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} \left( (\hat{Z}^k - US^k V^T)_{ij} - (X W^k)_{ij} \right)^2 + \frac{1}{\gamma} \|W^k\|_F^2.$$  \hspace{1cm} (5.13)

This is a quadratic optimization problem, so it is efficiently solvable via gradient descent. Let $R$ be the tensor of residuals, where $R^k = \hat{Z}^k - US^k V$. Given a warm start solution $W_0$, maximum number of gradient steps $G$, and step size $\nu > 0$, we denote this update compactly as

$$W \leftarrow \text{OneSided}(R, \Omega, X, W_0, \gamma, G, \nu),$$

which is detailed in Algorithm 3.

4. Iterate until the variables $W, U, S, V$ converge.

We are guaranteed to reach a stationary point because each step decreases the objective value. We express the steps of the complete algorithm $\text{TensorOneSided}$ in Algorithm 6.

5.1.7 Tensor Model with Noisy Two-Sided Information

In this section, we introduce a low rank model for tensor completion given noisy two-sided information, and we present the method $\text{TensorTwoSided}$. This model combines components from the $\text{Tensor}$ and $\text{TwoSided}$ models.
Algorithm 6 TensorOneSided

Data: Tensor $Z \in \mathbb{R}^{n \times m \times \ell}$, with
- known entries $\Omega = \{(i, j, k) : z_{ij}^k \text{ is known}\}$,
- missing entries $\Omega^c = \{(i, j, k) : z_{ij}^k \text{ is missing}\}$,
- and side information $X \in \mathbb{R}^{n \times p}$.

Input: Rank $r$, warm start $Z_0 \in \mathbb{R}^{n \times m \times \ell}$,
- max number of iterations $T \geq 1$,
- regularization parameter $\gamma > 0$,
- max number of gradient steps $G \geq 1$, step size $\nu > 0$.

Output: Locally optimal solution to problem (5.11).

Procedure:
Initialize $\hat{Z} \leftarrow Z_0$, $W \leftarrow 0$, $t \leftarrow 0$.
while $t < T$ do
    $R^k \leftarrow \hat{Z}^k - XW^k \quad \forall k$,
    $U \leftarrow \text{svds}(R(1), r)$,
    $V \leftarrow \text{svds}(R(2), r)$,
    $S^k \leftarrow U^T R^k V \quad \forall k$,
    $R^k \leftarrow \hat{Z}^k - US^k V^T \quad \forall k$,
    $W \leftarrow \text{OneSided}(R, \Omega, X, W, \gamma, G, \nu)$,
    $\hat{z}_{ij}^k \leftarrow (US^k V^T + XW^k)_{ij} \quad \forall (i, j, k) \in \Omega^c$,
    $t \leftarrow t + 1$.
end while
return Estimated values of $W, U, S, V$. 
Figure 5-4: Tensor model of a drug screening data set with noisy side information for both the patients and drugs. \( n \) is the number of patients, \( m \) is the number of drugs, \( \ell \) is the number of doses tested, \( r \) is the number of latent features, \( p \) is the number of observed patient features, and \( q \) is the number of observed drug features.

Let \( x_i \) be the features of row \( i \), and let \( y_j \) be the features of column \( j \). The generative model for \( z_{ij}^k \) is

\[
\hat{z}_{ij}^k = u_i^T S^k v_j + x_i^T W^k y_j,
\]

(5.14)

where \( u_i \) are latent features of row \( i \), \( v_j \) are latent features of row \( j \), \( S^k \) are weights of the latent features for slice \( k \), and \( W^k \) are weights of the observed features for slice \( k \). It follows that the model for the \( k \)th slice of the tensor is

\[
\hat{Z}^k = US^k V^T + XW^k Y^T,
\]

(5.15)

where \( U \in \mathbb{R}^{n \times r} \), \( V \in \mathbb{R}^{m \times r} \), \( S^1, \ldots, S^\ell \in \mathbb{R}^{r \times r} \), and \( W^1, \ldots, W^\ell \in \mathbb{R}^{p \times q} \) are learned from the data. In Figure 5-4, we show a diagram of this tensor model with two-sided information for a drug screening data set.

To find \( W, U, S, V \), we consider the following optimization problem:

\[
\min_{W,U,S,V} \sum_{k=1}^\ell \sum_{(i,j) \in \Omega_k} (z_{ij}^k - (US^k V^T + XW^k Y^T)_{ij})^2 + \frac{1}{\gamma} \|W^k\|_F^2,
\]

(5.16)

where \( \gamma \) is a regularization parameter. This formulation uses two parameters \( \gamma \) and \( r \) which we can learn via cross-validation. Instead of the Frobenius norm, it is also reasonable to consider a nuclear norm penalty on each matrix of coefficients \( W^k \), or add the constraint that \( W^k \) is low rank. We consider the Frobenius norm here because this formulation is very
close to formulation (5.11) so we can use a similar solution method.

In Appendix B.2, we provide details for an alternating optimization procedure to solve problem (5.16), which we refer to as the \texttt{TensorTwoSided} algorithm. This algorithm is identical to the \texttt{TensorOneSided} algorithm except for the update of $W$ in Step 3.

5.2 Simulated Data Experiments

In this section, we present computational experiments testing the proposed methods for tensor completion on simulated data. In Section 5.2.1 we describe the generation process for the simulated data sets. In Section 5.2.2 we present the experimental setup and the methods which are compared. In Section 5.2.3 we present the results from all of the simulated data experiments.

5.2.1 Simulated Data Sets

For this set of experiments, we generate complete tensors $Z \in \mathbb{R}^{200 \times 200 \times 10}$ with low slice rank. In particular, we suppose that:

$$Z^k = US^kV^T, \quad \forall k = 1, \ldots, 10,$$

where:

- $U \in \mathbb{R}^{200 \times 20}$: ground truth latent features of the rows,
- $S^k \in \mathbb{R}^{20 \times 20}$: ground truth weights for the $k$th slice,
- $V \in \mathbb{R}^{200 \times 20}$: ground truth latent features of the columns.

We suppose that all of the entries of $U, S^1, \ldots, S^k, V$ are independently identically distributed $\mathcal{N}(0, 1)$. In addition, we suppose that the matrices of row and column side information are given by:

$$X = U + \epsilon^1,$$

$$Y = V + \epsilon^2,$$
where:

- \( \varepsilon_1 \in \mathbb{R}^{200 \times 20} \): random noise for the row features,
- \( \varepsilon_2 \in \mathbb{R}^{200 \times 20} \): random noise for the column features.

We suppose that all of the entries of \( \varepsilon_1, \varepsilon_2 \) are independently identically distributed \( \mathcal{N}(0, \sigma) \), where \( \sigma \geq 0 \) is the standard deviation of the feature noise which we will vary.

### 5.2.2 Experimental Setup

In these experiments, we impute missing values in tensors of the form \( Z \in \mathbb{R}^{200 \times 200 \times 10} \) described in the previous section. For this task, we suppose that we are given the observed values of \( Z \) and side information \( X \in \mathbb{R}^{200 \times 20}, Y \in \mathbb{R}^{200 \times 20} \) for some level of noise \( \sigma \geq 0 \). In each experiment, we randomly select 80% of the values in the tensor to be missing completely at random (MCAR). We then compare a variety of methods for predicting these missing values in the tensor, including:

1. **Tensor**: Implements the **Tensor** method given in Algorithm 5 to impute the missing values via the Slice Learning method [51]. This method learns a low rank representation of the 3-dimensional data, including latent features for the rows and columns which are constant across all of the slices. Uses cross-validation to select the tensor rank \( r \).

2. **Two-Sided**: Implements the **TwoSided** method given in Algorithm 4 to impute the missing values for each slice independently via an \( \ell_2 \)-regularized bilinear regression model. Uses interaction terms between observed features of the rows and columns as features in the model. Uses cross-validation to select the regularization parameter \( \gamma \).

3. **Tensor Two-Sided**: Implements the **TensorTwoSided** method given in Algorithm 6 which incorporates both observed and latent features of the rows and columns. Uses cross-validation to select the tensor rank \( r \) with the weights of the side information \( W = 0 \) fixed. Then, with the optimal value of \( r \) fixed, uses cross-validation to select the regularization parameter \( \gamma \).
For each of the above methods, we tune the tensor rank $r$ over the range $\{1, 2, \ldots, 20\}$, and we tune the regularization parameter $\gamma$ over the range $\{0.1, 0.01, \ldots, 10^{-10}\}$. We evaluate the out-of-sample accuracy of each method and compare against a baseline which predicts the mean value of each tensor slice. For each method and missing data scenario, we compute the out-of-sample $R^2$ value on each slice, and then take the average of the out-of-sample $R^2$ values across all of the slices. We repeat all of the experiments 5 times varying the random seed which generates the ground truth tensor $Z \in \mathbb{R}^{200 \times 200 \times 10}$ and the missing data scenarios.

5.2.3 Results

In this section, we present the results from the experiments on simulated data.

In Figure 5-5, we plot the imputation accuracy of the tensor completion methods as we vary the standard deviation of the noise added to the side information. Across all levels of noise considered, the TensorTwoSided method significantly improves upon the next best method. As the level of noise increases, the performance of both TensorTwoSided and TwoSided decreases, while the performance of Tensor remains constant. At the highest noise level $\sigma = 1$, the average out-of-sample $R^2$ values were 0.957, 0.933, and 0.298 for the TensorTwoSided, Tensor, and TwoSided methods, respectively. This demonstrates that the proposed method TensorTwoSided can improve upon the baseline Tensor method even when the side information is only weakly predictive.

On the other hand, with no noise added ($\sigma = 0$), the average out-of-sample $R^2$ values were 0.997, 0.933, and 0.988 for the TensorTwoSided, Tensor, and TwoSided methods, respectively. This demonstrates that the proposed method TensorTwoSided can improve upon the baseline TwoSided regression method even when the row and column features are known exactly. Overall, these results show that the proposed method TensorTwoSided outperforms the best of the Tensor and TwoSided methods across all noise levels considered.
Figure 5-5: Imputation accuracy for the simulated data experiments with 80% missing data, varying the standard deviation of the normally distributed feature noise.

5.3 Real-world Data Experiments

In this section, we present computational experiments testing the proposed methods for tensor completion on two large-scale anti-cancer drug screens. In Sections 5.3.1 and 5.3.2, we describe the Genomics of Drug Sensitivity in Cancer (GDSC) and the Cancer Cell Line Encyclopedia (CCLE) data sets. In Section 5.3.3, we present the experimental setup and the methods which are compared. In Section 5.3.4, we present the results from all of the real-world data experiments.

5.3.1 Genomics of Drug Sensitivity in Cancer

The first anti-cancer drug screening data set that we consider is the Genomics of Drug Sensitivity in Cancer (GDSC) data set \[135\]. We are given data \(Z \in \mathbb{R}^{n \times m \times \ell}\) from experiments applying anti-cancer drugs to patients, where \(n = 955\), \(m = 265\), and \(\ell = 12\) are the numbers of patients, drugs, and doses, respectively. For each drug \(j\), the \(\ell\)th dose corresponds the maximum concentration at which drug \(j\) was administered, and the \(k\)th dose is \(1/2\) times the concentration of the \((k + 1)\)th dose for \(k = 1, \ldots, (\ell - 1)\). In addition, we have genomic data \(X \in \mathbb{R}^{n \times p}\) where \(p = 2,004\) is the number of genomic features. These features include
mutation, gain-loss, and whole exome sequence information for the oncogenes identified in the Catalogue of Somatic Mutations in Cancer (COSMIC) data set [53], as well as tissue type and cancer classification according to The Cancer Genome Atlas (TGCA) groupings [131].

5.3.2 Cancer Cell Line Encyclopedia data set

We also consider the Cancer Cell Line Encyclopedia (CCLE) anti-cancer drug screening data set [3]. In this data set, we are given data $Z \in \mathbb{R}^{n \times m \times \ell}$ from experiments applying anti-cancer drugs to patients, where $n = 461$, $m = 24$, and $\ell = 8$ are the numbers of patients, drugs, and doses, respectively. For all drugs, the $\ell$th dose corresponds the maximum concentration of 8$\mu$M, and the $k$th dose is approximately $1/3.2$ times the concentration of the $(k + 1)$th dose for $k = 1, \ldots, (\ell - 1)$. In addition, we have genomic data $X \in \mathbb{R}^{n \times p}$ where $p = 2,036$ is the number of genomic features. These features include copy number variation, mutation, and RNA expression data for the oncogenes identified in the COSMIC data set [53].

5.3.3 Experimental Setup

In these experiments, we impute missing values in tensors of drug sensitivity values from the GDSC [135] and CCLE [3] data sets. For each dose, we ignore the already missing values and hide an additional 20%, 40%, 60%, or 80% of the observed values to be the test set. We then compare a variety of methods for predicting these missing values in the tensor, including:

1. **Piecewise Linear**: Uses linear interpolation to fill in each missing (patient, drug, dose) response using the (patient, drug) responses that are available at the higher and lower doses. For (patient, drug) pairs with zero observations, this method imputes the mean of the drug response at that dose. This is a fast method that we use as a warm start for the other methods which require one.

2. **Non-Linear Mixed Effects (NLME)**: Uses a multilevel mixed effects model to simultaneously fit two-parameter sigmoidal dose response curves for all (patient, drug) pairs [126]. For each sigmoidal curve, the two free parameters are assumed to be
normally distributed about the mean values for the entire data set. Uses the Piecewise Linear imputation as a warm start.

3. **Matrix**: Fills in the missing values for each dose independently with matrix completion via SoftImpute \(^{[89]}\). Uses the Piecewise Linear imputation as a warm start and cross-validation to select the optimal matrix rank, which may be different for each slice of the tensor.

4. **Tensor**: Implements the Tensor method given in Algorithm \(^{[5]}\) to impute the missing values via the Slice Learning method \(^{[51]}\). This method learns a low rank representation of the 3-dimensional data, including latent features for the patients and drugs which are constant across all of the doses. Uses the Piecewise Linear imputation as a warm start and cross-validation to select the tensor rank \(r\).

5. **Genomic**: Implements the OneSided method given in Algorithm \(^{[3]}\) to impute the missing values for each dose independently via an \(\ell_2\)-regularized regression model. Uses genomic features of the patients as the row side information. Uses cross-validation to select the regularization parameter \(\gamma\).

6. **Tensor Genomic**: Implements the TensorOneSided method given in Algorithm \(^{[6]}\) which incorporates both genomic features of the patients and latent features of the patients and drugs. Uses cross-validation to select the tensor rank \(r\) with the weights of the side information \(W = 0\) fixed. Then, with the optimal value of \(r\) fixed, uses cross-validation to select the regularization parameter \(\gamma\). Uses the Piecewise Linear imputation as a warm start.

In the Matrix method, we tune the matrix ranks over the range \(\{1, 2, \ldots, 20\}\). For the Tensor and TensorOneSided methods, we tune the tensor rank \(r\) over the ranges \(\{10, 20, \ldots, 120\}\) for the GDSC data set and \(\{1, 2, \ldots, 20\}\) for the CCLE data set. For the TensorOneSided method, we tune the regularization parameter \(\gamma\) over the range \(\{0.1, 0.01, \ldots, 10^{-10}\}\).

We evaluate the out-of-sample accuracy of each method and compare against a baseline which predicts the mean value of each tensor slice. For each method and missing data
scenario, we compute the out-of-sample $R^2$ value on each slice, and then take the average of the out-of-sample $R^2$ values across all of the slices. We repeat all of the experiments 5 times varying the random seed which generates the missing data scenarios.

5.3.4 Results

In this section, we present the results from the real-world experiments on the anti-cancer drug screening data sets.

In Figures 5-6 and 5-7, we show the average out-of-sample $R^2$ for each method on the GDSC and CCLE data sets under different missing scenarios. For both sets of experiments, we see that the TensorGenomic method performs best in all missing percentages. As the percentage of missing data increases, the relative improvement over the Tensor method increases, while the relative improvement over the Genomic method decreases. This makes sense because as there is greater missing data in the tensor, the side information becomes more important.

On the GDSC data set, we see that Tensor and TensorGenomic are equally the best methods when there is 20-60% missing data, and TensorGenomic outperforms both Tensor and Genomic when there is 80% missing data. At low missing percentages, the third best method is NLME, which is the mixed-effects model to fit sigmoidal dose response curves that has been used in recent publications on the GDSC data set [126, 65]. However, at high missing percentages, the performance of the NLME method tails off considerably and its $R^2$ even turns negative. In contrast, the TensorGenomic, Tensor, Genomic, and Matrix methods all maintain $R^2$ values of 0.25 or greater. This indicates that matrix factorization-based and regression-based models can add value over current parametric models for fitting dose response curves, especially in scenarios with lots of missing data.

On the CCLE data set, we observe similar trends. One difference is that TensorGenomic outperforms Tensor in all missing percentages and matches Genomic as the best method with 80% missing data. In addition, the Genomic method is much stronger relative to other methods across the board. This suggests that the genomic features that we selected in the CCLE data set are more predictive than the genomic features that we selected in the
GDSC data set. As in the previous data set, the NLME method declines in performance rapidly as the percent of missing data increases, and is significantly outperformed by matrix factorization-based and regression-based models with 80% missing data.

We also present the tensor ranks which were selected during cross-validation for the tensor-based methods in Figures B-1 and B-2 in Appendix B.3. Since we select $r$ first during the cross-validation procedure for TensorGenomic, the rank parameters selected by both Tensor and TensorGenomic are the same in each experiment. In both data sets, the average tensor rank selected decreases as the percentage of missing data increases. In addition, the average tensor rank selected is much higher in the GDSC experiments than in the CCLE experiments, because the GDSC data set is much larger. This shows that we can fit more complicated tensor models (e.g. models with higher tensor ranks) when more data is available.
Figure 5-7: Imputation accuracy on the CCLE data set varying the percentage of missing data from 20% to 80%.

5.4 Discussion

In this section, we discuss the results from the experiments on simulated and real-world data in Sections 5.2 and 5.3.

Overall, both sets of experiments demonstrate that the proposed methods for tensor completion which combine a low rank and a regression component either match or outperform methods which have only one of these components. In the simulated data experiments, we see that the proposed method TensorTwoSided outperforms the low rank and regression methods across all levels of feature noise considered. In the real-world data experiments, we see that the proposed method TensorGenomic matches the low rank and regression methods across all percentages of missing data considered, and for each data set there is at least one missing percentage where it significantly improves upon both of the individual methods. For the GDSC data set, TensorGenomic strictly outperforms the other methods with 80% missing data, and for the CCLE data set, TensorGenomic strictly outperforms the other methods with 20%, 40%, and 60% missing data.

In addition, the computational experiments on real-world data show that the proposed methods outperform state-of-the-art methods for the task of predicting anti-cancer drug
response. First, we observe that the tensor model on its own significantly outperforms the multilevel mixed effects model which is used in practice. We suspect that the multilevel mixed effects model generalizes poorly because the dose response curves of some patients are significantly different from a “typical” sigmoidal dose response curve. Some patients may have mutations which make them completely resistant to certain anti-cancer drugs, while other patients may be extra sensitive to certain drugs. As a result, the dose response curves of these patients may be significantly different from the population average, which goes against the probabilistic assumptions of the multilevel mixed effects model.

Furthermore, the real-world experiments demonstrate that we can improve the out-of-sample performance of the tensor model using the genomic features which are available on the patients. We see that adding genomic data side information is more useful when the percentage of missing data is high. When the missing percentage is lower, most of the predictive power comes from the original tensor model. As a result, the final method TensorGenomic performs better than either the Tensor or Genomic methods individually.

These results suggest that the tensor data is quite valuable when it is available. One of the best predictors of an individual’s response to chemotherapy may be how this individual responded to previous rounds of chemotherapy, even at different drugs and doses. In a clinical setting, if a patient is receiving their 4th round of chemotherapy, we may be able to optimize the drug and dose depending on the results from their first 3 rounds of treatment along with their individual characteristics. However, if a patient is starting their first round of chemotherapy, then we must rely solely upon the individual characteristics to make a treatment decision.

5.5 Conclusions

In this chapter, we propose a new approach for tensor completion with noisy side information, and we introduce two methods which take into account noisy features of the rows and/or columns of the tensor, respectively. In computational experiments on real-world data sets, we show that the proposed method TensorGenomic works well in practice imput-
ing missing values in the GDSC and CCLE data sets leveraging genomic side information. For this particular application, our work demonstrates that tensor-based models are effective tools representing data from large-scale anti-cancer drug screens. More broadly, our work demonstrates that tensor-based models are powerful tools representing real-world data from complex systems, and these models can be easily augmented and improved with noisy side information.
Chapter 6

Conclusions

In this thesis, we have studied the problem of learning from missing and uncertain data with applications to personalized medicine. We have presented several new machine learning algorithms and have demonstrated their effectiveness on either benchmark machine learning data sets or particular learning tasks for personalized medicine. In this chapter, we describe the connections between the previous chapters and the major takeaways from this body of work considered as a whole.

In Part I, we found that the key idea of taking a rigorous optimization-based approach to classical problems in statistics works well for a wide range of problems. Computational experiments on an entire suite of benchmark data sets from the UCI machine learning repository provide empirical evidence for this in both Chapters 2 and 3. In addition, we saw that optimization is a powerful modeling tool which can readily accommodate diverse statistical models for learning. We leveraged this fact to consider a wide range of statistical models, including support vector machines (SVM), logistic regression, and decision trees for robust classification in Chapter 2 and K-nearest neighbors, SVM, and decision trees for optimal missing data imputation in Chapter 3.

In Part II, we honed in on developing algorithms for particular personalized medicine applications. Provided with specific structure of the data sets, we were able to develop more specialized algorithms to achieve significant performance gains on these learning tasks. These gains in accuracy have the potential to make real-world impact. In particular, the
MedImpute algorithm presented in Chapter 4 resulted in more accurate predictive models for 10-year risk of stroke and 60-day risk of mortality among late-stage cancer patients. The tensor completion algorithm presented in Chapter 5 resulted in more accurate prediction of anti-cancer drug response at particular dosages.

In addition, although the methods presented in Chapters 4 and 5 were developed with personalized medicine applications in mind, they could be used for other purposes as well. For example, MedImpute could be used to impute multivariate panel data from a store that collects information on its customers over time. An online retailer could use our tensor completion algorithm with noisy side information to predict customer behavior given click-thru, browsing, and purchase data. There is no shortage of real-world learning tasks with missing and uncertain data, and our work demonstrates that this is a promising direction for future research in machine learning.
Appendix A

Appendix for Chapter 2

This appendix includes supplementary material for Chapter 2 on Robust Classification. In Section A.1, we show the connection between classical SVM and feature-robust classification methods. In Section A.2, we provide derivations of the robust-in-both formulations for SVM, logistic regression, and decision trees that are introduced in Section 2.6.

A.1 Robustness Properties of Classical Support Vector Machines

The feature-robust counterpart presented in Theorem 1 is similar to the classical SVM problem (2.2). Making the substitution \( \tilde{\xi}_i = \xi_i - \rho \|w\|_q^* \) in Problem (2.11), we obtain

\[
\min_{w, b} n\rho \|w\|_q^* + \sum_{i=1}^n \tilde{\xi}_i
\]

s.t. \( y_i (w^T x_i - b) \geq 1 - \tilde{\xi}_i \quad \forall i, \]

\( \tilde{\xi}_i \geq -\rho \|w\|_q^* \quad \forall i. \) \hspace{1cm} (A.1)

Comparing Problem (A.1) to the classical SVM formulation (2.2), 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, [52] has shown that the robust optimization problem

\[
\min_{w, b} \max_{\Delta x \in \tilde{\mathcal{U}}_x} \sum_{i=1}^n \xi_i \\
\text{s.t. } y_i (w^T x_i - b) \geq 1 - \xi_i \quad \forall i, \\
\xi_i \geq 0 \quad \forall i, 
\]  

is exactly equivalent to the problem

\[
\min_{w, b} \rho \|w\|_q^* + \sum_{i=1}^n \xi_i \\
\text{s.t. } y_i (w^T x_i - b) \geq 1 - \xi_i \quad \forall i, \\
\xi_i \geq 0 \quad \forall i, 
\]  

where

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

It follows that (A.3) is equivalent to the classical SVM problem (2.2) for the choice of \( q^* = 2 \), or the \( \ell_1 \)-regularized SVM problem (2.3) 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.
A.2 Robust-in-Both Proofs

A.2.1 Soft-Margin Support Vector Machines

Proof of Theorem 7. Using a similar process as in the proof of Theorem 1, we rearrange the first constraint and solve the minimization problem. Problem (2.27) can be reformulated as

\[
\min_{w, b} \max_{\Delta y \in \mathcal{U}} \sum_{i=1}^{n} \xi_i \\
\text{s.t. } y_i(1 - 2\Delta y_i)((w^T x_i - b) \geq 1 - \xi_i + \rho \|w\|_{q^*}, \forall i, \\
\xi_i \geq 0 \quad \forall i.
\]

We can reformulate this as

\[
\min_{w, b} \max_{\Delta y \in \mathcal{U}} \sum_{i=1}^{n} \max\{1 - y_i(1 - 2\Delta y_i)((w^T x_i - b) + \rho \|w\|_{q^*}, 0\}.
\]

Now we follow the approach in the proof of Theorem 4. □

A.2.2 Logistic Regression

Proof of Theorem 8. Using a similar process as in the proof of Theorem 2, we first solve the innermost minimization problem and show that Problem (2.29) is equivalent to

\[
\max_{\beta, \beta_0} \min_{\Delta y \in \mathcal{U}} - \sum_{i=1}^{n} \log \left( 1 + e^{-y_i(1 - 2\Delta y_i)((\beta^T x_i + \beta_0) + \rho \|\beta\|_{q^*}} \right).
\]

Now we follow the approach in the proof of Theorem 5. Since the polyhedron \(\{\Delta y \in \mathbb{R}^n \mid \sum_{i=1}^{n} \Delta y_i \leq \Gamma, 0 \leq \Delta y_i \leq 1\}\) has integer extreme points, the inner minimization problem
above has the same objective as when the integer constraints are relaxed:

\[
\min_{\Delta y} - \sum_{i=1}^{n} \log \left( 1 + e^{-y_i(1 - 2\Delta y_i)(\beta^T x_i + \beta_0) + \rho \|\beta\|_{q^*}} \right)
\]

s.t. \( 0 \leq \Delta y_i \leq 1 \quad \forall i, \)

\[
\sum_{i=1}^{n} \Delta y_i \leq \Gamma.
\]

Define the function \( f_i(\Delta y_i) = \log \left( 1 + e^{-y_i(1 - 2\Delta y_i)(\beta^T x_i + \beta_0) + \rho \|\beta\|_{q^*}} \right) \) for \( i = 1, \ldots, n. \) Because \( \Delta y_i \in \{0, 1\}, \) we can express \( f_i(\Delta y_i) \) as

\[
f_i(\Delta y_i) = [f(1) - f(0)]\Delta y_i + f(0)
\]

\[
= \log \left( \frac{1 + e^{-y_i(\beta^T x_i + \beta_0) + \rho \|\beta\|_{q^*}}}{1 + e^{y_i(\beta^T x_i + \beta_0) + \rho \|\beta\|_{q^*}}} \right) \Delta y_i
\]

\[
- \log \left( 1 + e^{-y_i(\beta^T x_i + \beta_0) + \rho \|\beta\|_{q^*}} \right).
\]

The inner minimization problem can thus be expressed as

\[
\min_{\Delta y} \sum_{i=1}^{n} \left[ \log \left( \frac{1 + e^{-y_i(\beta^T x_i + \beta_0) + \rho \|\beta\|_{q^*}}}{1 + e^{y_i(\beta^T x_i + \beta_0) + \rho \|\beta\|_{q^*}}} \right) \Delta y_i
\]

s.t. \( 0 \leq \Delta y_i \leq 1 \quad \forall i, \)

\[
\sum_{i=1}^{n} \Delta y_i \leq \Gamma.
\]

By strong duality, the inner minimization problem has the same objective value as its dual problem. Replacing the inner minimization in Problem (A.4) with the dual problem yields the desired result.
A.2.3 Optimal Decision Trees

Proof of Theorem 9. Since the set of constraints affected by applying Theorem 3 and set of constraints affected by applying Theorem 6 are disjoint, we can simply apply them both simultaneously to yield the stated result. □

The full robust-in-both Optimal Tree formulation is therefore

\[
\begin{align*}
\min & \quad \sum_{k=1}^{K} f_k - \sum_{k=1}^{K} \lambda_k d_k \\
\text{s.t.} & \quad g_k = \sum_{i=1}^{n} \frac{1 - y_i}{2} z_{ik} \quad \forall k, \quad \text{(A.5a)} \\
& \quad h_k = \sum_{i=1}^{n} \frac{1 + y_i}{2} z_{ik} \quad \forall k, \quad \text{(A.5b)} \\
& \quad f_k \leq g_k - \Gamma \mu_{1,k} - \sum_{i=1}^{n} \nu_{1,ik} + M[w_k + (1 - c_k)] \quad \forall k, \quad \text{(A.5c)} \\
& \quad f_k \leq h_k - \Gamma \mu_{2,k} - \sum_{i=1}^{n} \nu_{2,ik} + M[(1 - w_k) + (1 - c_k)] \quad \forall k, \quad \text{(A.5d)} \\
& \quad f_k \geq g_k + \Gamma \mu_{3,k} + \sum_{i=1}^{n} \nu_{3,ik} - M[(1 - w_k) + (1 - c_k)] \quad \forall k, \quad \text{(A.5e)} \\
& \quad f_k \geq h_k + \Gamma \mu_{4,k} + \sum_{i=1}^{n} \nu_{4,ik} - M[w_k + (1 - c_k)] \quad \forall k, \quad \text{(A.5f)} \\
& \quad \mu_{m,k} + \nu_{m,ik} \geq -y_i z_{ik} \quad \forall i, k, \forall m \in \{1, 4\}, \quad \text{(A.5g)} \\
& \quad \mu_{m,k} + \nu_{m,ik} \geq y_i z_{ik} \quad \forall i, k, \forall m \in \{2, 3\}, \quad \text{(A.5h)} \\
& \quad d_k = 1 \quad \forall k \in \{\lceil K/2 \rceil, \ldots, K\}, \quad \text{(A.5i)} \\
& \quad d_k \leq d_j \quad \forall k, \forall j \in \mathcal{P}_k, \quad \text{(A.5j)} \\
& \quad d_k + \sum_{l=1}^{p} a_{kl} = 1 \quad \forall k, \quad \text{(A.5k)} \\
& \quad \sum_{k=1}^{K} z_{ik} = 1 \quad \forall i, \quad \text{(A.5l)} \\
& \quad z_{ik} \leq d_k \quad \forall i, k, \quad \text{(A.5m)}
\end{align*}
\]
\[ z_{ik} \leq 1 - d_j \quad \forall i, k, \forall j \in \mathcal{P}_k, \quad (A.5o) \]
\[ \sum_{i=1}^{n} z_{ik} \geq N c_k \quad \forall k, \quad (A.5p) \]
\[ c_k \geq d_k - \sum_{j \in \mathcal{P}_k} d_j \quad \forall k, \quad (A.5q) \]
\[ a_j^T x_i + \rho + \epsilon \leq b_j + (1 - z_{ik}) \quad \forall i, k, \forall j \in \mathcal{P}_k^l, \quad (A.5r) \]
\[ a_j^T x_i - \rho \geq b_j + (1 - z_{ik}) \quad \forall i, k, \forall j \in \mathcal{P}_k^l, \quad (A.5s) \]
\[ a_k \in \{0, 1\}^p \quad \forall k, \quad (A.5t) \]
\[ 0 \leq b_k \leq 1 \quad \forall k, \quad (A.5u) \]
\[ z_{ik}, w_k, c_k, d_k \in \{0, 1\} \quad \forall i, k, \quad (A.5v) \]
\[ \mu_{m,k}, \nu_{m,ik} \geq 0 \quad \forall i, k, m. \quad (A.5w) \]
Appendix B

Appendix for Chapter 5

This appendix includes supplementary material for Chapter 5 on Tensor Completion with Noisy Side Information. In Appendix B.1 we provide formal definitions of tensor rank. In Appendix B.2 we present the details of the TensorTwoSided algorithm to solve the tensor completion problem given noisy two-sided information which is presented in Section 5.1.7. In Appendix B.3 we provide plots of the tensor rank which is selected for the Tensor and TensorOneSided methods for the computational experiments in Section 5.3.

B.1 Definitions of Tensor Rank

In this section, we provide several definitions for the rank of a 3-dimensional tensor, including the CP rank, Tucker rank, and Slice rank. The definitions of CP rank and Tucker rank are well-known, and these are also described by [73]. The definition of Slice rank was introduced in recent work by [51].

1. **CP rank**: A tensor $Z \in \mathbb{R}^{n \times m \times \ell}$ is CP rank-1 if and only if it can be directly expressed as the outer product of vectors. In other words, there exists vectors $u \in \mathbb{R}^n$, $v \in \mathbb{R}^m$, $w \in \mathbb{R}^\ell$ such that $z_{ij}^k = u_i v_j w_k$ for all $i, j, k$. In general, the CP rank of a tensor $Z$ is the minimum number $r$ such that $Z$ can be expressed as the sum of $r$ CP rank-1 tensors.
2. **Tucker rank**: The Tucker rank is the tuple \((r_1, r_2, r_3)\) of column ranks of the mode-1, mode-2, and mode-3 unfoldings of the tensor, or equivalently:

\[
\text{Tucker}(Z) := (\text{rank}(Z_{(1)}), \text{rank}(Z_{(2)}), \text{rank}(Z_{(3)})).
\]

3. **Slice rank**: The slice rank is the maximum of the column ranks of the mode-1 and mode-2 unfoldings of the tensor, or equivalently:

\[
\text{Slice}(Z) := \max\{\text{rank}(Z_{(1)}), \text{rank}(Z_{(2)})\}.
\]

Further, if \(Z\) has Slice rank equal to \(r\), then we can find a decomposition such that \(Z^k = US^kV^T, k = 1, \ldots, \ell\) for some matrices \(U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{m \times r}\), and \(S^1, \ldots, S^\ell \in \mathbb{R}^{r \times r}\).

### B.2 TensorTwoSided Algorithm

In this section, we present the alternating minimization algorithm **TensorTwoSided** to solve the tensor completion problem given noisy two-sided information. This algorithm finds high-quality solutions to problem \((5.16)\) which was introduced in Section \([5.1.7]\). It is identical to the **TensorOneSided** algorithm except for the update of \(W\) in Step 3.

1. Begin with a warm start solution \(\hat{Z}\). Initialize all of the variables \(W, U, S, V\) to zero.

2. Update \(U, S, V\) by considering the following problem:

\[
\min_{U,S,V} \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} \left( (\hat{Z}^k - XW^kY^T)_{ij} - (US^kV^T)_{ij} \right)^2.
\]

We can find high-quality solutions to this problem using the Slice Learning algorithm \([51]\). Let \(R\) be the tensor of residuals, where \(R^k = \hat{Z}^k - XW^kY^T\). In this step, we find a low rank tensor approximation to \(R\) by taking SVDs of the mode-1 and mode-2 unfoldings.
3. Update the $W$ by considering the following problem:

$$\min_W \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} \left( (\hat{Z}^k - US^k V^T)_{ij} - (XW^k Y^T)_{ij} \right)^2 + \frac{1}{\gamma} \|W^k\|_F^2. \quad (B.2)$$

This is a quadratic optimization problem, so it is efficiently solvable via gradient descent. Let $R$ be the tensor of residuals, where $R^k = \hat{Z}^k - US^k V^T$. Given a warm start solution $W_0$, maximum number of gradient steps $G$, and step size $\nu > 0$, we denote this update compactly as

$$W \leftarrow \text{TwoSided}(R, \Omega, X, Y, W_0, \gamma, G, \nu),$$

which is detailed in Algorithm 4.

4. Iterate until the variables $W, U, S, V$ converge.

We are guaranteed to reach a stationary point because each step decreases the objective value. We express the steps of the complete algorithm $\text{TensorTwoSided}$ in Algorithm 7.
Algorithm 7 TensorTwoSided

**Data:** Tensor $Z \in \mathbb{R}^{n \times m \times \ell}$, with
- known entries $\Omega = \{(i, j, k) : z_{ij}^k \text{ is known}\}$,
- missing entries $\Omega^c = \{(i, j, k) : z_{ij}^k \text{ is missing}\}$,
- and side information $X \in \mathbb{R}^{n \times p}$, $Y \in \mathbb{R}^{m \times q}$.

**Input:** Rank $r$, warm start $Z_0 \in \mathbb{R}^{n \times m \times \ell}$,
- regularization parameter $\gamma > 0$,
- max number of iterations $T \geq 1$.

**Output:** Locally optimal solution to problem (5.16).

**Procedure:**

Initialize $\hat{Z} \leftarrow Z$, $W \leftarrow 0$, $t \leftarrow 0$.

while $t < T$ do

Rearrange $R^k \leftarrow \hat{Z}^k - X W^k Y^T$ $\forall k$,

$U \leftarrow \text{svds}(R_{(1)}, r)$,

$V \leftarrow \text{svds}(R_{(2)}, r)$,

$S^k \leftarrow U^T R^k V$ $\forall k$,

$R^k \leftarrow \hat{Z}^k - U S^k V^T$ $\forall k$,

$W \leftarrow \text{TwoSided}(R, \Omega, X, Y, W, \gamma, G, \nu)$,

$z_{ij}^k \leftarrow (U S^k V^T + X W^k Y^T)_{ij}$ $\forall (i, j, k) \in \Omega^c$,

t $\leftarrow t + 1$.
end while

return Estimated values of $W, U, S, V$. 

202
B.3 Plots of Cross-validated Tensor Rank

In this section, we provide plots of the average cross-validated tensor rank selected by the Tensor and TensorOneSided methods in the computational experiments in Section 5.3.

Figure B-1: Average Slice rank for the Tensor model on the GDSC data set at varying missing percentages. In each experiment, the rank is selected via cross-validation from the range \{10, 20, \ldots, 120\}.

Figure B-2: Average Slice rank for the Tensor model on the CCLE data set at varying missing percentages. In each experiment, the rank is selected via cross-validation from the range \{1, 2, \ldots, 20\}.
Bibliography


