Journal of Machine Learning Research 18 (2018) 1-39 Submitted 2/17; Revised 3/18; Published 4/18

From Predictive Methods to Missing Data Imputation: An Optimization Approach

Dimitris Bertsimas DBERTSIM@MIT.EDU
Colin Pawlowski CPAWLOWS@MIT.EDU
Ying Daisy Zhuo ZHUO@MIT.EDU
Sloan School of Management and Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139

Editor: Francis Bach

Abstract

Missing data is a common problem in real-world settings and for this reason has attracted significant attention in the statistical literature. We propose a flexible framework based on formal optimization to impute missing data with mixed continuous and categorical variables. This framework can readily incorporate various predictive models including K-nearest neighbors, support vector machines, and decision tree based methods, and can be adapted for multiple imputation. We derive fast first-order methods that obtain high quality solutions in seconds following a general imputation algorithm `opt.impute` presented in this paper. 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.

Keywords: missing data imputation, K-NN, SVM, optimal decision trees

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.

Method Name	Category	Software	Reference
Mean impute (`mean`)	Mean		Little and Rubin (1987)
Expectation-Maximization (`EM`)	EM		Dempster et al. (1977)
EM with Mixture of Gaussians and Multinomials	EM		Ghahramani and Jordan (1994)
EM with Bootstrapping	EM	`Amelia II`	Honaker et al. (2011)
K-Nearest Neighbors (`knn`)	K-NN	`impute`	Troyanskaya et al. (2001)
Sequential K-Nearest Neighbors	K-NN		Kim et al. (2004)
Iterative K-Nearest Neighbors	K-NN		Caruana (2001); Brás and Menezes (2007)
Support Vector Regression	SVR		Wang et al. (2006)
Predictive-Mean Matching (`pmm`)	LS	`MICE`	Buuren and Groothuis-Oudshoorn (2011)
Least Squares	LS		Bø et al. (2004)
Sequential Regression Multivariate Imputation	LS		Raghunathan et al. (2001)
Local-Least Squares	LS		Kim et al. (2005)
Sequential Local-Least Squares	LS		Zhang et al. (2008)
Iterative Local-Least Squares	LS		Cai et al. (2006)
Sequential Regression Trees	Tree	`MICE`	Burgette and Reiter (2010)
Sequential Random Forest	Tree	`missForest`	Stekhoven and Bühlmann (2012)
Singular Value Decomposition	SVD		Troyanskaya et al. (2001)
Bayesian Principal Component Analysis	SVD	`pcaMethods`	Oba et al. (2003); Mohamed et al. (2009)
Factor Analysis Model for Mixed Data	FA		Khan et al. (2010)

Table 1: List of Imputation Methods

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 (Little and Rubin, 1987). 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 (Breiman et al., 1984; Wagstaff, 2004).

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. We focus on methods for missing data imputation in this paper.

Concretely, assume that we are given data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with missing entries $x_{id}, (i,d) \in \mathcal{M}$. The objective is to impute the values of the missing data that resemble the underlying complete data as closely as possible. This way, when one conducts statistical inference or pattern recognition using machine learning methods on the imputed data, the results should be similar to those obtained if full data were given. We outline some of the state-of-the-art methods for imputation in Table 1 and describe them briefly below. Part of the list is adapted from a review paper by Liew et al. (2011).

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 (Little and Rubin, 1987).

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 (Honaker et al., 2011). Joint modeling provides useful theoretical properties but lacks the flexibility for processing data types seen in many real applications (Van Buuren, 2007). 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 (Ghahramani and Jordan, 1994).

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 (Van Buuren, 2007). To generalize to multivariate settings, a chained equation process — initializing using random sampling and conducting univariate imputations sequentially until convergence — is typically used (Buuren and Groothuis-Oudshoorn, 2011). 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 (Bø et al., 2004; Raghunathan et al., 2001). Alternatively, in the Predictive-Mean Matching method (`pmm`), imputations are random samples drawn from a set of observed values close to regression predictions (Buuren and Groothuis-Oudshoorn, 2011). Imputation methods that use Support Vector Regression in place of LS for the regression step have also been explored (Wang et al., 2006).

When there is non-linear relationship between the variables, linear regression based imputation may perform poorly. Burgette and Reiter (2010) propose using Classification and Regression Trees (CART) as the conditional model for imputation. Extensions to random forests have also shown promising results (Stekhoven and Bühlmann, 2012). These decision tree based imputation methods are non-parametric approaches that do not rely upon distributional assumptions on the data.

One of the most commonly used non-parametric approaches is K-Nearest Neighbors (K-NN) based imputation. This method imputes each missing entry x_{id} as the mean of the dth dimension of the K-nearest neighbors that have observed values in dimension d (Troyanskaya et al., 2001). 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 (Kim et al., 2004). Iterative K-NN uses an iterative process to refine the estimates and choose the nearest neighbors based on the estimates from the previous iteration (Caruana, 2001; Brás and Menezes, 2007). The Local-Least Squares method combines ideas from K-NN and LS, imputing each missing value x_{id} using regression models trained on the K-nearest neighbors of the point

$\mathbf{x}_i$ (Kim et al., 2005). Sequential and iterative variations of Local-Least Squares resemble their K-NN imputation counterparts (Zhang et al., 2008; Cai et al., 2006).

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 (Troyanskaya et al., 2001; Mazumder et al., 2010). Bayesian Principal Component Analysis is similar to SVD imputation but extends the method to incorporate information from a prior distribution on the model parameters (Oba et al., 2003; Mohamed et al., 2009). Some recent development of a variant of the EM algorithm for factor analysis also provides a missing data imputation method for mixed data (Khan et al., 2010).

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 (Little and Rubin, 1987). 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 (Buuren and Groothuis-Oudshoorn, 2011). 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 (Honaker et al., 2011). Some conditional specification methods such as `pmm` are practically reliable, but lack theoretical foundation and have no explicit formulation as an optimization problem. This stands in stark contrast to other areas of machine learning, where statistical models and optimization problems are deeply intertwined.

Evidence from recent literature suggests that recent advances in optimization have driven significant progress in machine learning. Integer and convex optimization have been applied successfully to median and sparse regression problems (Bertsimas and Van Parys, 2017; Bertsimas and Mazumder, 2014). Recent work on Optimal Decision Trees for classification leverages integer and robust optimization (Bertsimas and Dunn, 2017; Bertsimas et al., 2017). In this paper, we reconsider the missing data problem from this perspective, in order to develop optimization-based methods for imputation with improved out-of-sample performance.

1.2 Contributions

We summarize our contributions in this paper 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 `opt.impute` presented in this paper. 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, `opt.impute` produces the best overall imputation in more than 75.8% of all data sets, and results in an average reduction in mean absolute error of 8.3% against the best cross-validated benchmark method.

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

The structure of the paper is as follows. In Section 2, we formulate the missing data imputation problem as an optimization problem, present a general first-order method `opt.impute` that can be used to find high-quality solutions, and derive the algorithms for each model: K-NN, SVM, and trees. We also discuss a cross-validation procedure and extensions of `opt.impute` to multiple imputation. In Section 3, we compare the imputation quality and performance on downstream tasks of `opt.impute` to benchmark imputation methods on a wide range of real data sets. In Section 4, we discuss the benefits from adopting such framework and suggest areas for future work. We conclude in Section 5.

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 dth categorical variable $d \in \{p_0 + 1, \ldots, p_0 + p_1\}$ takes values among k_d classes. Note that if all data is continuous $p_0 = 0$, while if all data is categorical $p_1 = 0$. The missing and known values are specified by the following sets:

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

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

2.1 General Problem Formulation

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

$$
\begin{aligned}
\min \quad & c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X}) \\
\text{s.t.} \quad & w_{id} = x_{id} \quad (i,d) \in \mathcal{N}_0, \\
& v_{id} = x_{id} \quad (i,d) \in \mathcal{N}_1, \\
& (\mathbf{U}, \mathbf{W}, \mathbf{V}) \in \mathcal{U},
\end{aligned}
\tag{1}
$$

where $\mathcal{U}$ is the set of all feasible combinations $(\mathbf{U}, \mathbf{W}, \mathbf{V})$ of auxiliary vectors and imputations. For example, in a K-NN based approach, this includes the constraints that each point has exactly K neighbors and the assignment variables are binary. We list the auxiliary variables and cost functions corresponding to each of the imputation models K-NN, SVM, and trees in Table 2. Note that the cost function can be different for continuous and categorical variables. We can introduce a parameter that controls the relative contribution to the cost between the continuous and categorical variables, or scale continuous variables appropriately. For the remainder of the paper the latter is assumed for simplicity of notation.

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

Table 2: Variables and cost functions for each imputation model. Variables for K-NN, SVM, and trees are defined in Sections 2.3, 2.4, and 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 `opt.impute` which approximates the solution to Problem (1) very fast using first-order methods.

2.2 First-Order Method for the General Problem

To obtain high-quality solutions to Problem (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) (Bertsekas, 1999). Algorithm 1, which we refer to as `opt.impute`, implements BCD or CD for Problem (1). The variables $\mathbf{U}, \mathbf{W}, \mathbf{V}$, and $\mathbf{X}$ as well as the cost function $c(\cdot)$ are summarized in Table 2 for K-NN, SVM, and trees. The detailed solution methods for Problems (2), (3), (4), and (5) for K-NN, SVM, and tree imputation models are described in Sections 2.3-2.5, respectively.

By construction, the objective function value strictly decreases by at least δ_0 until termination. It follows that the number of steps needed for the algorithm to terminate is $\lceil \frac{1}{\delta_0} c(\mathbf{U}^0, \mathbf{W}^0, \mathbf{V}^0; \mathbf{X}) \rceil$, where $\mathbf{W}^0, \mathbf{V}^0$ are the initialization values, $\mathbf{X}$ is data, and $\mathbf{U}^0$ is the argmin in Equation (2). However, the algorithm is not guaranteed to find a global minimum for Problem (1) (Wright, 2015).

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

Algorithm 1 `opt.impute`

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

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

Procedure:

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

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

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

$$\begin{aligned} \mathbf{U}^* \leftarrow \arg\min_{\mathbf{U}} \quad & c(\mathbf{U}, \mathbf{W}^{old}, \mathbf{V}^{old}; \mathbf{X}) \\ \text{s.t.} \quad & (\mathbf{U}, \mathbf{W}^{old}, \mathbf{V}^{old}) \in \mathcal{U}. \end{aligned} \tag{2}$$

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

2a block coordinate descent (BCD):

$$\begin{aligned} \mathbf{W}^*, \mathbf{V}^* \leftarrow \arg\min_{\mathbf{W},\mathbf{V}} \quad & c(\mathbf{U}^*, \mathbf{W}, \mathbf{V}; \mathbf{X}) \\ \text{s.t.} \quad & w_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_1, \\ & (\mathbf{U}^*, \mathbf{W}, \mathbf{V}) \in \mathcal{U}, \end{aligned} \tag{3}$$

or

2b coordinate descent (CD):

$$\begin{aligned} w^*_{jr} \leftarrow \arg\min_{w_{jr}} \quad & c(\mathbf{U}^*, \mathbf{W}, \mathbf{V}; \mathbf{X}) \\ \text{s.t.} \quad & w_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_1, \\ & w_{id} = w^*_{id} \qquad (i,d) \in \mathcal{M}_0 \backslash (j,r), \\ & v_{id} = v^*_{id} \qquad (i,d) \in \mathcal{M}_1, \\ & (\mathbf{U}^*, \mathbf{W}, \mathbf{V}) \in \mathcal{U}, \end{aligned} \tag{4}$$

$$\begin{aligned} v^*_{jr} \leftarrow \arg\min_{v_{jr}} \quad & c(\mathbf{U}^*, \mathbf{W}, \mathbf{V}; \mathbf{X}) \\ \text{s.t.} \quad & w_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} \qquad (i,d) \in \mathcal{N}_1, \\ & w_{id} = w^*_{id} \qquad (i,d) \in \mathcal{M}_0, \\ & v_{id} = v^*_{id} \qquad (i,d) \in \mathcal{M}_1 \backslash (j,r), \\ & (\mathbf{U}^*, \mathbf{W}, \mathbf{V}) \in \mathcal{U}. \end{aligned} \tag{5}$$

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

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

end while

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

for $\mathbf{U}, \mathbf{W}, \mathbf{V}$ that we use in our optimization-based imputation procedure. After, we describe a cross-validation procedure to select the specific model and parameters for the imputation.

2.3 K-NN Based Imputation

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

$$d_{ij} := \sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} \mathbb{1}_{\{v_{id} \neq v_{jd}\}}. \tag{6}$$

Next, we introduce the binary variables:

$$z_{ij} = \begin{cases} 1, & \text{if } (\mathbf{w}_j, \mathbf{v}_j) \text{ is among the } K\text{-nearest neighbors of } (\mathbf{w}_i, \mathbf{v}_i) \\ & \text{with respect to distance metric (6),} \\ 0, & \text{otherwise.} \end{cases}$$

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:

$$\begin{aligned} \min \quad & c(\mathbf{Z}, \mathbf{W}, \mathbf{V}; \mathbf{X}) := \sum_{i \in \mathcal{I}} \sum_{j=1}^{n} z_{ij} \Big[\sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} \mathbb{1}_{\{v_{id} \neq v_{jd}\}} \Big] \\ \text{s.t.} \quad & w_{id} = x_{id} && (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} && (i,d) \in \mathcal{N}_1, \\ & z_{ii} = 0 && i \in \mathcal{I}, \\ & \sum_{j=1}^{n} z_{ij} = K && i \in \mathcal{I}, \\ & \mathbf{Z} \in \{0,1\}^{|\mathcal{I}| \times n}. \end{aligned} \tag{7}$$

By optimality, it follows that $z_{ij} = 1$ if and only if $(\mathbf{w}_j, \mathbf{v}_j)$ is among the K-nearest neighbors of $(\mathbf{w}_i, \mathbf{v}_i)$. Therefore, solving Problem (7) produces the missing value imputation which minimizes the sum of distances from each point $(\mathbf{w}_i, \mathbf{v}_i), i \in \mathcal{I}$ to its K-nearest neighbors. Note that the relation $\mathbb{1}_{\{v_{id} \neq v_{jd}\}}$ can be modeled with binary variables. Problem (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 `opt.knn`.

2.3.1 opt.knn

In step ①, to update the auxiliary variables $\mathbf{Z}$, first fix all imputed values $\mathbf{W}$, $\mathbf{V}$. Problem (2) decomposes by $i \in \mathcal{I}$ into the assignment problems:

$$\begin{aligned} \min_{\mathbf{z}_i} \quad & \sum_{j=1}^{n} z_{ij} d_{ij} \\ \text{s.t.} \quad & z_{ii} = 0, \\ & \sum_{j=1}^{n} z_{ij} = K, \\ & \mathbf{z}_i \in \{0,1\}^n. \end{aligned} \tag{8}$$

The optimal solution to Problem (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 $(\mathbf{w}_i, \mathbf{v}_i)$ and set $z_{ij} = 1$ for these neighbors, $z_{ij} = 0$, otherwise.

Next, we fix $\mathbf{Z}$ and update the imputed values $\mathbf{W}, \mathbf{V}$ using either BCD or CD. In step (2a), the BCD update, Problem (3) decomposes by dimension $d = 1, \ldots, p$. For each continuous dimension $d = 1, \ldots, p_0$, we consider the following quadratic optimization problem:

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

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

$$(K + \sum_{j \in \mathcal{I}} z_{ji}) w_{id} - \sum_{(j,d) \in \mathcal{M}_0} (z_{ij} + z_{ji}) w_{jd} - \sum_{(j,d) \in \mathcal{N}_0} (z_{ij} + \mathbb{1}_{\{j \in \mathcal{I}\}} z_{ji}) x_{jd} = 0. \tag{9}$$

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

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

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

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

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

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

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

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

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

Solving the above gives

$$w_{id} = \frac{\sum_{j=1}^{n} z_{ij} w_{jd} + \sum_{j\in\mathcal{I}} z_{ji} w_{jd}}{K + \sum_{j\in\mathcal{I}} z_{ji}}. \tag{10}$$

We can interpret the missing value imputation (10) as a weighted average of the K nearest neighbors of $\mathbf{x}_i$, along with all points $\mathbf{x}_j$ which include $\mathbf{x}_i$ as a neighbor. Similarly, each categorical variable $v_{id}, (i,d) \in \mathcal{M}_1$ is imputed as the minimizer of the following:

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

The solution is

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

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

2.4 Mixed SVM Based Imputation

In this section, we consider a second model for imputation, based upon SVM regression for imputing continuous features and SVM classification for imputing categorical features. First, define $\widetilde{\mathbf{v}}_i \in \{-1,1\}^{p_2}$ to be a dummy encoded representation of $\mathbf{v}_i$, where $p_2 = \sum_{d=p_0+1}^{p_0+p_1} k_d - p_1$. Let $\widetilde{v}_{id}^{fixed}, (i,d) \in \mathcal{N}_2$ be the known dummy encoded values. For each continuous feature $d \in \{1,\dots,p_0\}$, let $(\boldsymbol{\beta}_d, \beta_{d0}) \in \mathbb{R}^{p_0+p_2+1}$ be the coefficients for an SVM regression model regressing feature d on the other features with the dummy encoding. Let $(\boldsymbol{\theta}_d, \theta_{d0}) \in \mathbb{R}^{p_0+p_2+1}$ be the coefficients for an SVM classification model predicting dummy feature d based upon the other features. Note that it is also possible to use a multi-class SVM model to predict each categorical feature directly, as described by Crammer and Singer (2001), using parameters of the form $\mathbf{M} \in \mathbb{R}^{k_d \times (p_0+p_2+1)}$ for each feature $d \in \{p_0+1,\dots,p_0+p_1\}$. In this case, we would keep the dummy encoded decision variables as covariates to predict the other features and add constraints relating $v_{id}, (i,d) \in \mathcal{M}_1$ and $\widetilde{v}_{id}, (i,d) \in \mathcal{M}_2$. For illustrative purposes and simplicity of notation, we present the formulation using binary SVM to predict each dummy variable d.

We consider the following optimization problem:

$$
\begin{aligned}
\min \quad & c([\boldsymbol{\beta},\boldsymbol{\theta}],\mathbf{W},\widetilde{\mathbf{V}};\mathbf{X}) := \frac{1}{2}\left(\|\boldsymbol{\theta}\|^2 + \|\boldsymbol{\beta}\|^2\right) + C\left(\sum_{i=1}^{n}\sum_{d=1}^{p_0}(\gamma_{id}+\gamma_{id}^*) + \sum_{i=1}^{n}\sum_{d=p_0+1}^{p_0+p_1}\xi_{id}\right) \\
\text{s.t.} \quad & x_{id} = w_{id} && (i,d) \in \mathcal{N}_0, \\
& \widetilde{v}_{id} = \widetilde{v}_{id}^{fixed} && (i,d) \in \mathcal{N}_2, \\
& \beta_{dd} = 0 && d = 1,\dots,p_0, \\
& \theta_{dd} = 0 && d = 1,\dots,p_2, \\
& \gamma_{id} \geq w_{id} - \left(\boldsymbol{\beta}_d^T \begin{bmatrix}\mathbf{w}_i \\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \beta_{d0}\right) - \epsilon && d = 1,\dots,p_0, i = 1\dots,n, \\
& \gamma_{id}^* \geq \left(\boldsymbol{\beta}_d^T \begin{bmatrix}\mathbf{w}_i \\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \beta_{d0}\right) - w_{id} - \epsilon && d = 1,\dots,p_0, i = 1\dots,n, \\
& \xi_{id} \geq 1 - \widetilde{v}_{id}\left(\boldsymbol{\theta}_d^T \begin{bmatrix}\mathbf{w}_i \\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \theta_{d0}\right) && d = 1,\dots,p_2, i = 1\dots,n, \\
& \gamma_{id} \geq 0 && d = 1,\dots,p_0, i = 1\dots,n, \\
& \gamma_{id}^* \geq 0 && d = 1,\dots,p_0, i = 1\dots,n, \\
& \xi_{id} \geq 0 && d = 1,\dots,p_2, i = 1\dots,n, \\
& \widetilde{v}_{id} \in \{-1,1\} && d = 1,\dots,p_2, i = 1\dots,n.
\end{aligned}
\tag{11}
$$

This formulation is based upon SVM with a linear kernel; however we can extend Problem (11) to arbitrary kernels, including the multi-class cases, using the modified objective function

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

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

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

2.4.1 `opt.svm`

In step 1, we fix the imputed values $\mathbf{W}, \mathbf{V}$ and update the auxiliary variables $[\boldsymbol{\beta}, \boldsymbol{\beta}_0, \boldsymbol{\theta}, \boldsymbol{\theta}_0]$. Independent of the choice of kernel, Problem (2) decomposes by dimension p into p_0 SVM regression problems and p_2 SVM classification problems for the categorical variables. For each continuous feature $d \in \{1, \ldots, p_0\}$, we update $\boldsymbol{\beta}_d, \beta_{d0}$ by solving

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}(\gamma_{id} + \gamma_{id}^*) \\
\text{s.t.} \quad & \beta_{dd} = 0 \\
& \gamma_{id} \geq w_{id} - (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - \epsilon \quad && i = 1 \ldots, n, \\
& \gamma_{id}^* \geq (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon \quad && i = 1 \ldots, n, \\
& \gamma_{id} \geq 0 && i = 1, \ldots, n, \\
& \gamma_{id}^* \geq 0 && i = 1, \ldots, n.
\end{aligned}
\tag{12}
$$

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

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{i=1}^{n}\xi_{id} \\
\text{s.t.} \quad & \theta_{dd} = 0 \\
& \xi_{id} \geq 1 - \widetilde{v}_{id}(\boldsymbol{\theta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \theta_{d0}) \quad && i = 1 \ldots, n, \\
& \xi_{id} \geq 0 && i = 1, \ldots, n.
\end{aligned}
\tag{13}
$$

Taking the Lagrangian duals, both Problems (12) and (13) can be reformulated as quadratic optimization problems which can be solved efficiently (Cortes and Vapnik, 1995).

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

$$
\begin{aligned}
\min_{\mathbf{w}_i, \widetilde{\mathbf{v}}_i} \quad & \sum_{d=1}^{p_0} (\gamma_{id} + \gamma_{id}^*) + \sum_{d=p_0+1}^{p_0+p_1} \xi_{id} \\
\text{s.t.} \quad & x_{id} = w_{id} && (i,d) \in \mathcal{N}_0, \\
& \gamma_{id} \geq w_{id} - (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - \epsilon && d = 1, \ldots, p_0, \\
& \gamma_{id}^* \geq (\boldsymbol{\beta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon && d = 1, \ldots, p_0, \\
& \xi_{id} \geq 1 - \widetilde{v}_{id} (\boldsymbol{\theta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \theta_{d0}) && d = 1, \ldots, p_2, \\
& \gamma_{id} \geq 0 && d = 1, \ldots, p_0, \\
& \gamma_{id}^* \geq 0 && d = 1, \ldots, p_0, \\
& \xi_{id} \geq 0 && d = 1, \ldots, p_2,
\end{aligned}
\tag{14}
$$

where $(\mathbf{w}_i, \widetilde{\mathbf{v}}_i) \in \mathbb{R}^{p_0} \times \{-1, 1\}^{p_2}$ is the imputation for observation $\mathbf{x}_i$. Note that if all features are continuous, Problem (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 (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:

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

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

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

2.5 Tree Based Imputation

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

For general prediction tasks, we can use greedy (Breiman et al., 1984) or globally optimal (Bertsimas and Dunn, 2017) solution methods to train the decision trees. In this case, we consider the latter approach because it admits a clear optimization model with mixed integer decision variables which fits into our framework for imputation. For each dimension d, let $\mathbf{T}^d \in \{0,1\}^{n \times n}$ denote the set of indicator variables

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

Let $(\mathbf{T}^d, \mathbf{W}, \mathbf{V}) \in \mathcal{T}^d$ denote the set of optimal decision tree constraints for dimension d as described in (Bertsimas and Dunn, 2017). We consider the following optimization problem:

$$\begin{aligned} \min \quad & c(\mathbf{T}, \mathbf{W}, \mathbf{V}; \mathbf{X}) := \sum_{i=1}^n \sum_{j=1}^n \Big[\sum_{d=1}^{p_0} \; t_{ij}^d (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} t_{ij}^d \mathbb{1}_{\{v_{id} \neq v_{jd}\}} \Big] \\ \text{s.t.} \quad & w_{id} = x_{id} \qquad\qquad (i,d) \in \mathcal{N}_0, \\ & v_{id} = x_{id} \qquad\qquad (i,d) \in \mathcal{N}_1, \\ & (\mathbf{T}^d, \mathbf{W}, \mathbf{V}) \in \mathcal{T}^d \qquad d = 1, \dots, p, \end{aligned} \tag{15}$$

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

2.5.1 `OPT.TREE`

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

Next, we fix $\mathbf{T}$ and update the imputed values $\mathbf{W}, \mathbf{V}$ using BCD or CD. In step 2a, Problem (3) decomposes by dimension into p_0 quadratic optimization problems and p_1

integer optimization problems. For each continuous dimension $d = 1, \dots, p_0$, we solve:

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

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

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

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

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

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

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

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

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

$$\min_{w_{id}} \quad 2\sum_{j=1}^{n} t_{ij}^d (w_{id} - w_{jd})^2. \tag{16}$$

An optimal solution to Problem (16) is

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

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

$$\min_{v_{id}} \quad 2 \sum_{j=1}^{n} t_{ij}^d \mathbb{1}_{\{v_{id} \neq v_{jd}\}}. \tag{17}$$

An optimal solution to Problem (17) is

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

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

2.6 Model Selection Procedure

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

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

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

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

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

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

The hyperparameters that we tune via this method are summarized in Table 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 3.1 for descriptions of these individual methods).

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

Table 3: Hyperparameters tuned via the model selection procedure outlined in Section 2.6. σ^2 is a parameter in the radial basis function kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma^2})$. cp is a complexity parameter related to the depth of the decision tree.

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 (Little and Rubin, 1987). 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. 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.

3.1 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 10.

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

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

2. **Predictive-Mean Matching** (`pmm`): An iterative method which imputes missing values from known values in a given dimension using linear regressions. It is commonly used for multiple imputation and can be generalized to multiple missing dimensions using the chained equations process (Buuren and Groothuis-Oudshoorn, 2011). Implemented using the `MICE` package in R.

3. **Bayesian PCA** (`bpca`): A missing data estimation method based on Bayesian principal component analysis (Oba et al., 2003). 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 (Brás and Menezes, 2007; Caruana, 2001) .

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 (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 (11) using a radial basis function kernel. For continuous variables, we use ϵ-support vector regression; for categorical variables, we use classical support vector machines. These problems were solved using coordinate descent methods. The implementation was in Julia using the `scikit-learn` package in Python.

(c) Decision Tree based (`opt.tree`): This method solves the optimal decision-tree problem (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 (Bertsimas and Dunn, 2017). We run coordinate descent for the imputation problems. The implementation was in Julia using the packages `glmnet` and `OptimalTrees`.

In addition, we consider two composite methods: `opt.cv`, which selects the best method from `opt.knn`, `opt.svm`, and `opt.tree`; and `benchmark.cv`, which selects the best method from `mean`, `pmm`, `bpca`, `knn`, and `iknn`. These composite methods use the cross-validation procedure described in Section 2.6. To generate the validation set for each missing data problem, we randomly sample an additional 10% of the entries to be hidden under the MCAR assumption. After running each individual method, we select the one that gives the lowest MAE on the validation set. We re-run this method on the original missing data set to obtain the final imputation.

Each imputation method was run for a maximum time limit of 12 hours on each data set. The quality of the imputations is evaluated using the same MAE and RMSE metrics defined in Section 2.6. For each of the `opt.impute` methods, we also record and present the convergence in objective value and MAE to show the progress over the iterations.

3.1.1 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 4. The MCAR assumption implies that the missing pattern is completely independent from both the missing and observed values. The NMAR assumption implies that the missing pattern depends upon the missing values. There is an intermediate type of assumption, missing at random (MAR), which implies that the missing pattern depends only upon the observed values, but not upon the missing values. Because this assumption is less general than NMAR, we do not consider this mechanism for our experiments.

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

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

Table 4: Statistical assumptions of mechanisms used to generate patterns of missing data $\mathcal{M}$ for data set $\mathbf{X}$. Here, we suppose that f is the underlying density of the missing pattern, and $\mathbf{X}^{obs}, \mathbf{X}^{miss}$ are the observed and missing components of the data set, respectively.

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.

3.1.2 Downstream Tasks

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

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

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

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

2. **Optimal Impute for Multiple Imputation** (`opt.mi`): Starting from m warm starts, we run `opt.knn`, `opt.svm`, or `opt.tree` to generate a new set of m fully

imputed data sets. We use warm starts produced by `mice`, and the best model among K-NN, SVM, and trees is selected initially via cross-validation.

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

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.

3.2.1 Convergence

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

3.2.2 Imputation Accuracy

The imputation accuracy for each data set is presented in Table 10 for the scenario in which 30% of the entries are missing, assuming MCAR. We compare the benchmark ones and each individual `opt.impute` method (not cross-validated); the method with the lowest

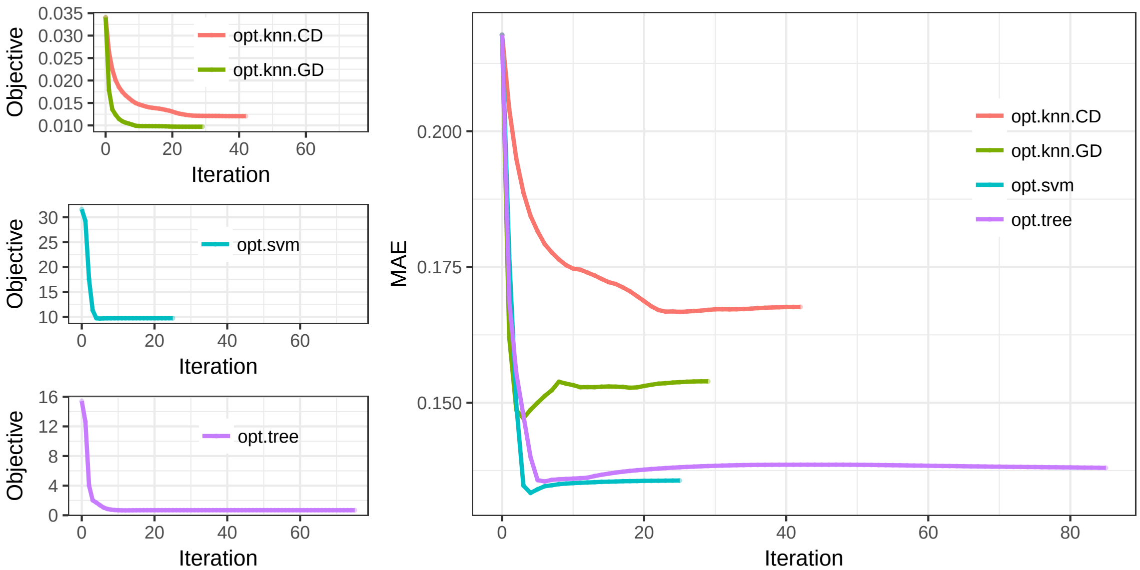


Figure 1: Solution paths of `opt.impute` methods on the `iris` data set. These plots show the objective value and mean absolute error (MAE) of the imputation over the course of the algorithm. Each path represents a different algorithm: `opt.knn` (BCD and CD), `opt.svm` (CD), and `opt.tree` (CD). Mean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

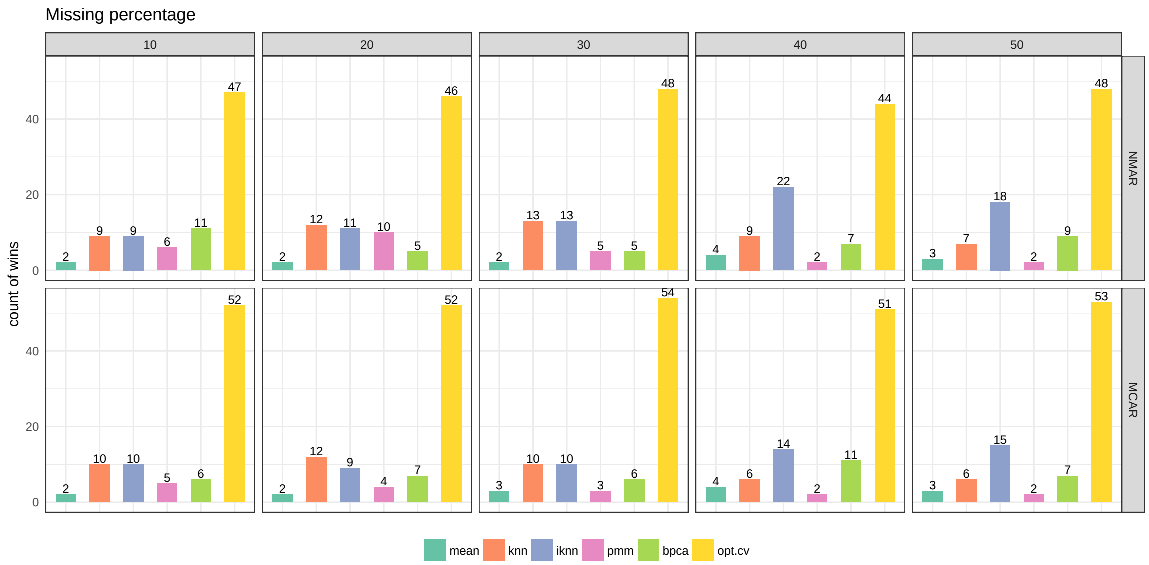
MAE (i.e., best imputation accuracy) is bolded. Among all data sets, at least one of the `opt.impute` methods obtains the lowest MAE in 76.2% of the data sets, followed by `iknn` and `bpca` imputation methods with 9 and 4 wins each. Comparatively, `mean`, `knn`, and `pmm` impute have the weakest performances.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

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

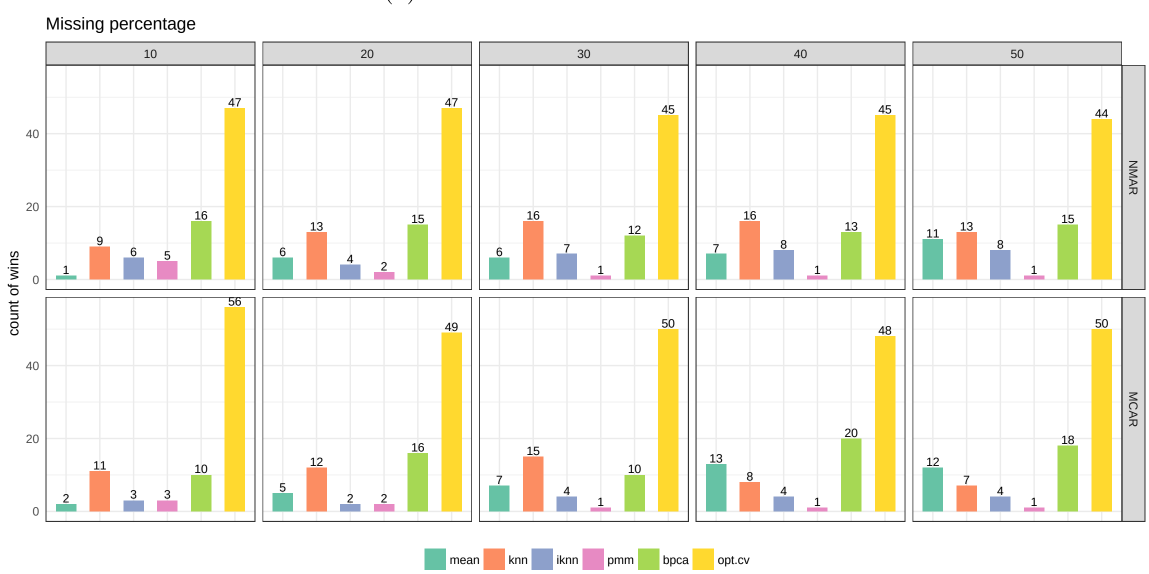
In Figure 3, we present summary results of the MAE and RMSE values as geometric means across all data sets for each missing percentage and missing data mechanism, with the confidence bands representing one geometric standard deviation multiplied above and divided below by the mean. Comparatively, `opt.cv`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

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 5 presents the statistical comparisons between each `opt.impute` method and each benchmark method. We conduct pairwise Wilcoxon signed rank tests and paired t-tests between each pair of methods. When comparing `opt.cv` against the benchmark methods, our proposed cross-validated method achieves statistically significant lower rank and lower MAE compared to each benchmark. For each individual `opt.impute` method, with the exception of `opt.svm` against heuristic `iknn`, the `opt.impute` one has statistically significant lower rank than every benchmark. The decrease in MAE is still statistically significant when `mean`, `bpca`, and `pmm` are comparators, but no longer statistically significant when compared to `knn` or `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.

Finally, we compare against the same cross-validated procedure introduced in Section 2.6 applied on all the benchmark methods (`benchmark.cv`) with results in Figure 2b. At 30% missing data, we observe 10.1% average improvement in MAE down to 0.118 from 0.131.



(a) Counts based on lowest MAE



(b) Counts based on lowest RMSE

Figure 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 5: Pairwise Wilcoxon signed-rank tests and t-tests between `opt.impute` and benchmark methods, with the p-values adjusted for multiple comparisons.

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

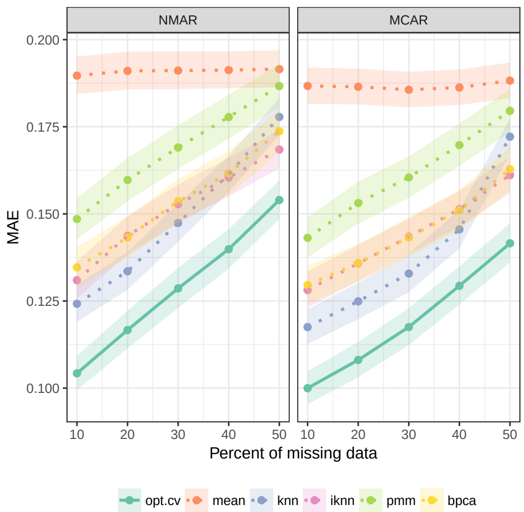
Further, `opt.cv` achieves highest imputation accuracy in more than 78.6% of the data sets compared to `benchmark.cv`.

3.2.3 Performance on Downstream Tasks

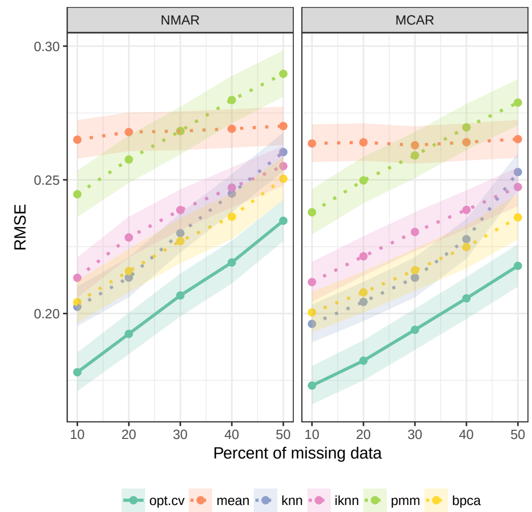
Next, we evaluate the performance of standard machine learning algorithms for classification and regression trained on the imputed data. We consider the data sets in Table 6, which were selected as a representative subsample from the UCI Machine Learning Repository data sets. These data sets range in 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 sets, the downstream tasks become more difficult as the missing data percentage increases.

In Figure 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 7 and 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.

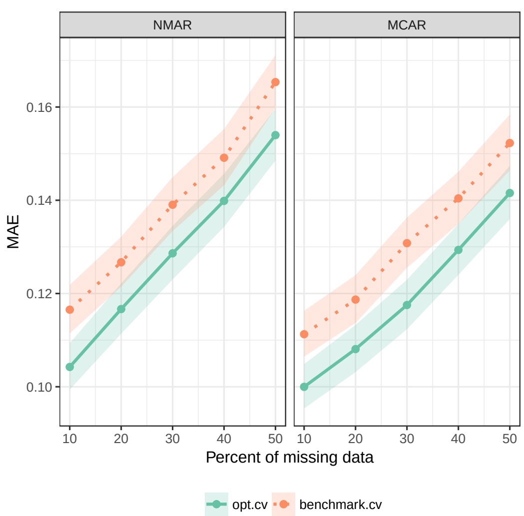
For the single imputation methods, we observe that the improvement of `opt.cv` over the best cross-validated benchmark method is statistically significant for all missing percentages in both classification and regression tasks. Moreover, this improvement in out-of-sample



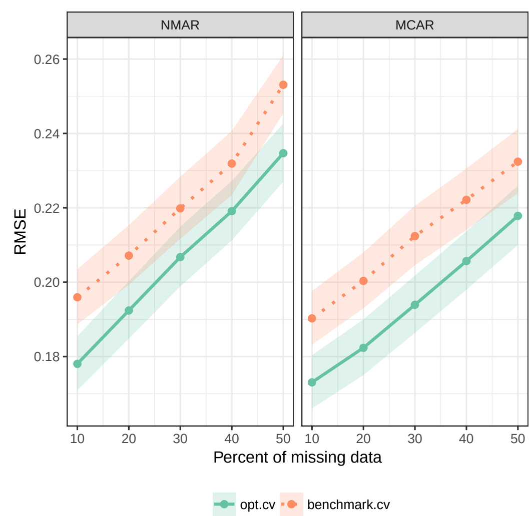
(a) Average MAE



(b) Average RMSE



(c) Average RMSE



(d) Average RMSE

Figure 3: Mean absolute error (MAE) and root mean squared error (RMSE) across 84 data sets for each imputation method, comparing `opt.cv` against all benchmark methods and against the cross-validated best benchmark method, `benchmark.cv`. The center lines are geometric mean with one geometric standard deviation multiplied above and divided below. The x-axis corresponds to the percentage of missing entries.

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

Table 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.

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.

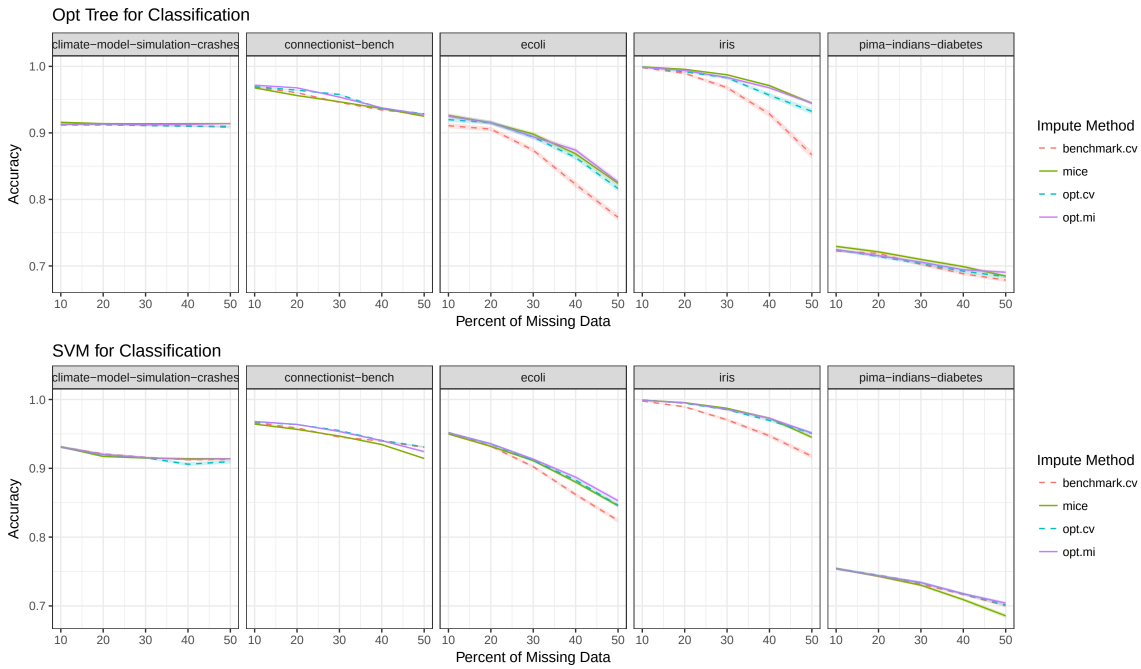
For the multiple imputation methods, we observe that the improvement of `opt.mi` over `mice` is statistically significant for all missing percentages in the regression tasks, and 3/5 missing percentages in the classification tasks. At the 50% missing percentage, the average improvement is 0.5% in out-of-sample 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.

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

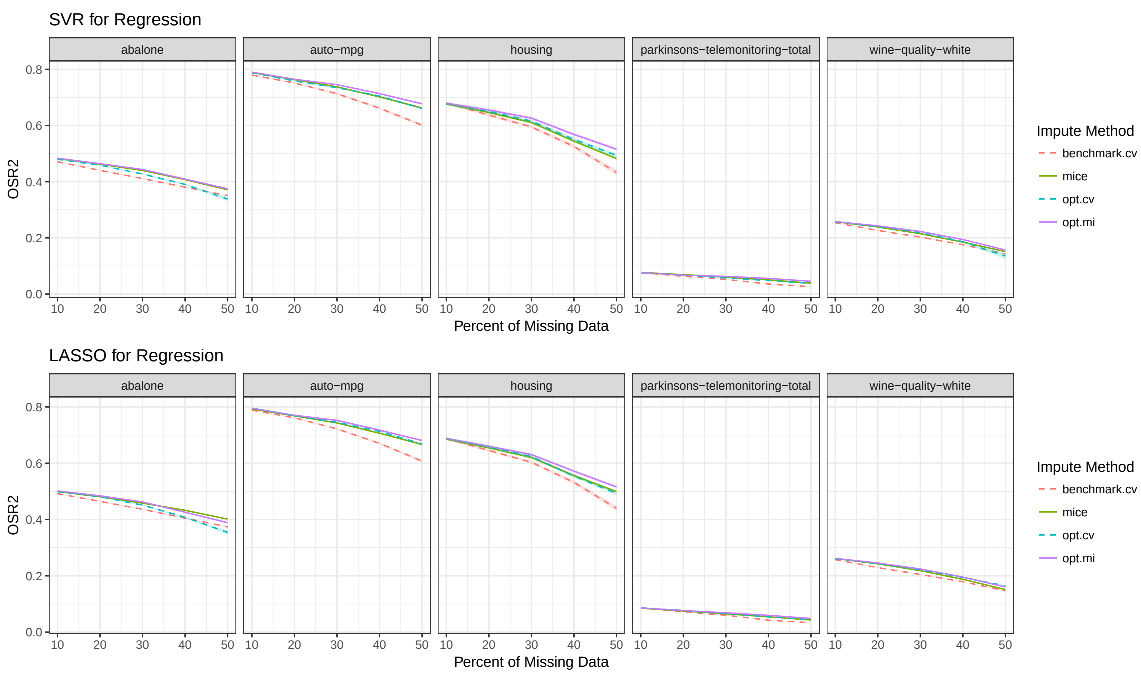
Finally, we compare the performance of single vs multiple imputation for `opt.impute`. We observe that the improvement of `opt.mi` over `opt.cv` is statistically significant in 8/10 scenarios, with the largest improvements occurring at the highest missing percentages. At the 50% missing percentage, the 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 `mice`.

3.2.4 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 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



(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 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.

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

Table 7: Pairwise t-tests between `opt.impute` and benchmark methods for downstream classification tasks, with the p-values adjusted for multiple comparisons.

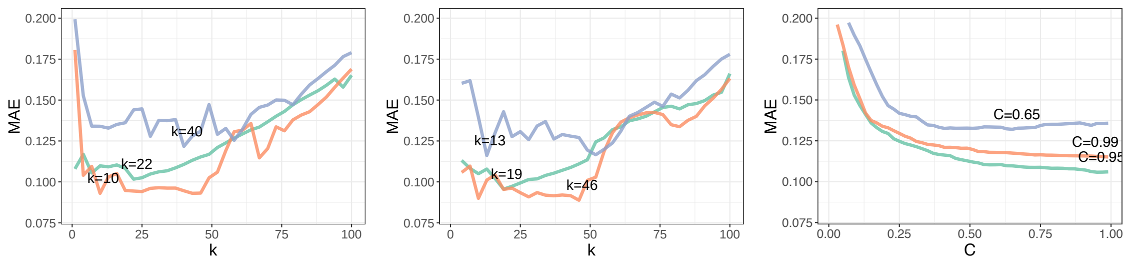
	Δ Out-of-Sample R^2 (adjusted p-value)		
Missing %	`opt.mi - mice`	`opt.cv - benchmark.cv`	`opt.mi - opt.cv`
10	0.0014 (<0.001***)	0.0034 (<0.001***)	0.0013 (<0.001***)
20	0.0029 (<0.001***)	0.0113 (<0.001***)	0.0027 (<0.001***)
30	0.0071 (<0.001***)	0.0161 (<0.001***)	0.0077 (<0.001***)
40	0.0085 (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

Table 8: Pairwise t-tests between `opt.impute` and benchmark methods for downstream regression tasks, with the p-values adjusted for multiple comparisons.

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 5 shows the relationship between the hyperparameters and MAE for various data sets and missing patterns. For `opt.knn` (CD and BCD), the out-of-sample MAE first decreases and then increases as the hyperparameter increases. When K reaches the sample size, the imputation is equivalent to mean imputation. For `opt.svm`, the imputation accuracy remains relatively constant with respect to changes in parameter C after a certain threshold. There were no external parameters for trees, as the trees in each step were pruned during the training process. Overall, these plots suggest that the `opt.impute` methods are relatively robust even if their hyperparameters are not known exactly.

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



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

Figure 5: Sensitivity of MAE to the choice of K for the number of neighbors for K-NN coordinate descent, K-NN block coordinate descent, and the trade-off parameter C for SVM in data set `iris`. The colors represent different missing data percentages. The parameter value that achieves lowest MAE is labeled for each missing data percentage.

3.2.5 Computational Speed

Next, we compare the computational time required for all imputation methods across a selection of six UCI data sets and missing data patterns. Each method was run on a single thread of a machine with an Intel Xeon CPU E5-2650 (2.00 GHz) Processor and limited to 8 GB RAM with a time limit of 4 hours. For various `opt.impute` methods, we report the running times for `mean` warm starts, as multiple warm starts can be trivially parallelized. The results are shown below in Table 9.

Mean imputation is almost instantaneous and is therefore not presented in the table. For small-scale problems on the `iris` data set, all imputation methods finish quickly. As the data dimension p increases (for example, in the `libras-movement` data set), most `opt.impute` methods scale better than the `pmm` method. As the sample size n increases, `opt.knn.CD` also scales better than `pmm`, as seen in `banknote-authentication` and `skin-segmentation`. Among the `opt.impute` methods, tree based imputation scales very well with respect to sample size n but not dimension p. Despite its high imputation quality, SVM based imputation scales relatively poorly with respect to both n and p. Among the proposed methods, `opt.knn.CD` has the best scalability in both n and p.

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

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

Table 9: Computational time comparison of benchmark and `opt.impute` imputation methods. Blank entries indicate that the method failed to converge with the 4 hour time limit.

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.

4. Discussion

One of the primary contributions of this paper 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 paper, 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 (1) fast and accurately for any of the three examples of non-convex, non-linear cost functions $c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X})$ proposed in this paper remains an open question.

5. Conclusions

In summary, we frame the classical missing data problem as a non-convex optimization problem based upon a variety of predictive models. We propose a family of new imputation methods, `opt.impute`, which finds high quality solutions to this problem using fast first-order methods. Through extensive computational experiments on 84 data sets from the UCI Machine Learning Repository, we show that `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.

Acknowledgments

The authors thank the reviewers who provided many insightful comments which improved the final manuscript. The research of the second author was supported by a National Science Foundation Predoctoral Fellowship.

Name	n	p_0	p_1	Benchmark mean	pmm	bpca	knn	iknn	opt.impute knn	svm	tree
acute-inflammations-1	120	1	5	0.3701	0.3626	0.2307	0.2694	0.3598	0.2285	0.2267	**0.2185**
acute-inflammations-2	120	1	5	0.3701	0.3626	0.2307	0.2694	0.3598	0.2285	0.2267	**0.2185**
airfoil-self-noise	1503	5	0	0.2332	0.2270	0.2332	0.2018	0.2054	**0.1944**	0.1949	0.2002
airline-costs	31	9	0	0.1799	0.1566	0.1054	0.1113	0.1071	**0.0970**	0.1084	0.1037
auto-mpg	392	5	2	0.2404	0.1793	0.1547	0.1623	0.1690	0.1396	0.1362	**0.1291**
balance-scale	625	4	0	**0.3011**	0.4112	**0.3011**	0.3503	0.3113	0.3701	0.3206	0.3049
banknote-authentication	1372	4	0	0.1608	0.1596	0.1608	0.1321	0.1361	**0.1117**	0.1182	0.1243
beer-aroma	23	8	0	0.2036	0.2004	0.1772	0.1773	0.1838	0.1728	0.1638	**0.1628**
blood-transfusion	748	4	0	0.1123	0.1215	0.1123	0.0945	0.0880	0.0799	0.0824	**0.0664**
breast-cancer-diagnostic	569	30	0	0.1066	0.0431	0.0558	0.0520	0.0565	0.0486	0.0512	**0.0351**
breast-cancer-prognostic	194	31	1	0.1304	0.0727	0.0850	0.0846	0.0911	0.0794	0.0682	**0.0576**
breast-cancer	683	8	1	0.2458	0.1531	**0.1318**	0.1541	0.1788	0.1367	0.1355	0.1333
climate-model-crashes	540	18	0	**0.2505**	0.3404	**0.2505**	0.2651	0.2570	0.2750	0.2921	0.2519
communities-and-crime-2	111	101	23	0.1374	0.2191	0.1137	0.0864	0.1053	0.0845	0.0875	**0.0577**
communities-and-crime	123	99	23	0.1613	0.2901	0.1327	0.0987	0.1252	0.0973	0.0936	**0.0711**
computer-hardware	209	7	1	0.1989	0.1888	0.1989	0.1824	**0.1703**	0.1917	0.1780	0.1832
concrete-compressive	103	7	0	0.2338	0.2005	0.2057	0.2053	0.1982	0.1854	0.1868	**0.1750**
concrete-flow	103	7	0	0.2338	0.2005	0.2057	0.2053	0.1982	0.1854	0.1868	**0.1750**
concrete-slump	103	7	0	0.2338	0.2005	0.2057	0.2053	0.1982	0.1854	0.1868	**0.1750**
congressional-voting-records	232	0	16	0.4357	0.4351	0.2150	0.2504	0.4357	**0.2107**	0.2449	0.3509
connectionist-bench-sonar	208	60	0	0.1629	0.1208	0.1440	0.1088	0.1219	0.1071	0.0918	**0.0905**
connectionist-bench	990	10	0	0.1506	0.1632	0.1294	0.1049	0.1001	**0.0829**	0.1143	0.1224
construction-maintenance	33	4	0	0.3614	**0.2461**	0.3638	0.3299	0.2836	0.3283	0.3250	0.3979
contraceptive-method-choice	1473	8	1	0.2767	0.2768	0.2519	0.2634	0.2336	**0.2229**	0.2263	0.2452
dermatology	358	33	1	0.2254	0.1447	0.1484	0.1212	0.1421	**0.1082**	0.1364	0.1957
diabetes	43	2	0	0.1868	0.2768	0.1868	**0.1844**	0.2095	0.2404	0.1847	0.1950
ecoli	336	7	0	0.1215	0.1224	0.0938	0.1071	0.0908	0.0990	0.1109	**0.0904**
fertility	100	7	2	0.3526	0.3854	0.3433	0.3432	0.3476	**0.3369**	0.3450	0.3665
flags	194	22	6	0.3246	0.3146	0.3246	0.2542	0.3039	**0.2475**	0.3290	0.2603
geographic-origin	1059	68	0	0.0827	0.0764	0.0599	0.0510	0.0557	0.0477	0.0584	**0.0438**
glass-identification	214	9	0	0.1140	**0.0825**	0.0956	0.0862	0.0865	0.0851	0.0923	0.0862
haberman-survival	306	3	0	0.1701	0.2258	0.1701	0.1754	**0.1663**	0.1734	0.1727	0.1696
hayes-roth	132	4	0	**0.2768**	0.3719	0.2778	0.2873	0.2779	0.2965	0.2948	0.2770
heart-disease-cleveland	297	8	5	0.3261	0.3386	0.2878	0.2945	0.3023	0.2763	**0.2738**	0.3041
hepatitis	80	4	15	0.3094	0.3019	0.3094	0.2753	0.2626	**0.2573**	0.2657	0.3480
hill-valley-noise	606	100	0	0.0998	0.0105	0.0066	0.0052	0.0283	**0.0051**	0.0781	0.0114
hill-valley	606	100	0	0.0971	0.0974	0.0055	0.0042	0.0273	0.0042	0.0783	**0.0031**
housing	506	13	0	0.1821	0.1211	0.1154	0.0985	0.1042	**0.0798**	0.1049	0.1261
hybrid-price	153	3	0	0.1538	0.1605	0.1538	0.1289	**0.1069**	0.1370	0.1202	0.1231
image-segmentation	210	19	0	0.1450	0.0806	0.0856	0.0637	0.0672	**0.0627**	0.0846	0.0628
immigrant-salaries	35	3	0	0.2247	0.2134	0.2247	0.1869	0.1700	0.1901	**0.1673**	0.1808
indian-liver-patient	579	8	2	0.1039	0.0953	0.0954	0.0981	0.0873	0.0910	0.1167	**0.0789**
ionosphere	351	34	0	0.2016	0.1739	0.1552	**0.1107**	0.1187	0.1172	0.1206	0.1475
iris	150	4	0	0.2200	0.1292	0.1571	0.1274	0.1370	0.1132	**0.1048**	0.1130
japan-emmigration	45	5	0	0.2096	0.2625	0.2098	0.2064	**0.1737**	0.2097	0.1866	0.2131
lenses	24	0	4	0.6607	0.6667	0.6696	**0.6339**	0.6607	0.6786	0.6786	0.6667
libras-movement	360	90	0	0.1823	0.0304	0.1022	0.0670	0.1014	0.0688	0.0522	**0.0139**
lpga-2008	157	6	0	0.1459	0.1769	0.1424	0.1448	0.1414	0.1496	**0.1294**	0.1299
lpga-2009	146	11	0	0.1750	0.1048	0.1074	0.1169	0.1131	0.1047	0.0889	**0.0881**
lung-cancer	27	0	56	0.3677	0.3475	0.3644	0.3426	0.3677	0.3586	**0.3348**	0.3438
mammographic-mass	830	0	5	0.2803	0.3307	0.2691	0.2386	0.2762	0.3390	0.2439	**0.2243**
monks-problems-1	124	0	6	0.6441	0.6396	0.6441	0.6059	0.6441	0.6411	**0.5991**	0.6502
monks-problems-2	169	0	6	0.6405	0.6373	0.6454	**0.6340**	0.6405	0.6481	0.6438	0.6383
monks-problems-3	122	0	6	0.6554	**0.5976**	0.6554	0.6813	0.6554	0.6577	0.6877	0.6622
parkinsons-telemonitoring-motor	5875	16	0	0.0623	0.0395	0.0372	0.0389	0.0342	0.0301	0.0458	**0.0265**
parkinsons-telemonitoring-total	5875	16	0	0.0623	0.0395	0.0372	0.0389	0.0342	0.0301	0.0458	**0.0265**
parkinsons	195	21	0	0.1348	0.0888	0.0849	0.0754	0.0814	**0.0690**	0.0824	0.0691
pima-indians-diabetes	768	8	0	0.1217	0.1453	0.1109	0.1164	0.1098	0.1089	**0.1049**	0.1069
planning-relax	182	12	0	0.1441	0.0823	0.1143	0.1188	0.1195	0.1019	0.0809	**0.0680**
post-operative-patient	87	0	8	0.3891	0.4428	0.3891	0.4143	**0.3861**	0.3937	0.4348	0.3955
pyrim	74	27	0	0.1798	0.1235	0.1758	0.1172	0.1193	**0.1145**	0.1219	0.1282
qsar-biodegradation	1055	41	0	0.0749	0.0379	0.0656	0.0385	0.0410	**0.0324**	0.0566	0.0452

Table 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.

				Benchmark					opt.impute		
Name	n	p_0	p_1	mean	pmm	bpca	knn	iknn	knn	svm	tree
seeds	210	7	0	0.2082	0.0795	0.0651	0.1099	0.0862	0.0715	0.0730	**0.0644**
soybean-large	266	0	35	0.2880	0.2583	0.2467	0.1874	0.2880	**0.1858**	0.1865	0.2103
soybean-small	47	0	35	0.2689	0.2816	0.2673	0.1577	0.2689	**0.1571**	**0.1571**	0.1837
spect-heart	80	0	22	0.2173	0.2134	0.2083	0.1899	0.2173	0.1951	**0.1869**	0.1913
spectf-heart	80	44	0	0.1307	0.1631	0.1307	0.1226	0.1195	0.1141	**0.1058**	0.1138
statlog-project-landsat-satellite	4435	36	0	0.1556	0.0405	0.0472	0.0390	0.0480	0.0329	0.0345	**0.0293**
teaching-assistant-evaluation	151	1	4	0.4017	0.4074	0.4094	0.3711	0.3992	0.4086	0.5131	**0.3370**
thoracic-surgery	470	3	13	0.1469	0.1704	**0.1388**	0.1433	0.1463	0.1415	0.2205	0.1397
thyroid-disease-ann-thyroid	3772	21	0	0.0773	0.0774	0.0869	0.0723	**0.0603**	0.0838	0.1162	0.0729
thyroid-disease-new-thyroid	215	5	0	0.0935	0.1083	0.0887	0.0849	**0.0754**	0.0774	0.0893	0.0851
triazines	186	60	0	0.1574	0.0667	0.1184	0.0503	0.0708	**0.0454**	0.0892	0.0495
tv-sales	31	8	0	0.2073	0.1949	0.1808	0.1934	**0.1729**	0.1952	0.1731	0.1964
vote-for-clinton	2704	9	0	0.0644	0.0715	0.0538	0.0676	0.0552	**0.0523**	0.0633	0.0537
wall-following-robot-2	5456	2	0	0.0721	0.0955	0.0721	0.0754	0.0720	0.0792	0.0847	**0.0717**
wall-following-robot-24	5456	4	0	0.0917	0.1172	0.0917	0.0886	0.0872	**0.0862**	0.0946	0.0895
wiki4he	176	0	44	0.2200	0.2234	0.1857	0.1872	0.1968	0.1777	**0.1731**	0.2085
wine-quality-red	1599	11	0	0.0976	0.0945	0.0761	0.0796	0.0744	**0.0683**	0.0757	0.0742
wine-quality-white	4898	11	0	0.0756	0.0782	0.0668	0.0771	0.0645	0.0598	0.0676	**0.0597**
wine	178	13	0	0.1680	0.1537	0.1203	0.1184	0.1144	**0.1091**	0.1105	0.1296
yacht-hydrodynamics	308	6	0	0.2102	0.1991	0.2088	0.1858	0.1861	0.1866	0.1867	**0.1799**
yeast	1484	8	0	0.0721	0.0917	0.0689	0.0740	**0.0671**	0.0683	0.0928	0.0680
zoo	101	1	15	0.2892	0.2832	0.1835	0.1518	0.2860	0.1502	0.3637	**0.1478**

Table 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.

References

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

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

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

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

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

Trond Hellem Bø, Bjarte Dysvik, and Inge Jonassen. LSimpute: accurate estimation of missing values in microarray data with least squares methods. *Nucleic Acids Research*, 32(3):e34–e34, 2004.

Lígia P Brás and José C Menezes. Improving cluster-based missing value estimation of DNA microarray data. *Biomolecular Engineering*, 24(2):273–282, 2007.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees.* CRC press, 1984.

Lane F Burgette and Jerome P Reiter. Multiple imputation for missing data via sequential regression trees. *American Journal of Epidemiology*, 172:1070–1076, 2010.

Stef Buuren and Karin Groothuis-Oudshoorn. MICE: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3), 2011.

Zhipeng Cai, Maysam Heydari, and Guohui Lin. Iterated local least squares microarray missing value imputation. *Journal of Bioinformatics and Computational Biology*, 4(05): 935–957, 2006.

Rich Caruana. A non-parametric EM-style algorithm for imputing missing values. In *AISTATS*, 2001.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(Dec):265–292, 2001.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38, 1977.

Zoubin Ghahramani and Michael I Jordan. Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems*, pages 120–127, 1994.

James Honaker, Gary King, Matthew Blackwell, et al. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47, 2011.

Mohammad E Khan, Guillaume Bouchard, Kevin P Murphy, and Benjamin M Marlin. Variational bounds for mixed-data factor analysis. In *Advances in Neural Information Processing Systems*, pages 1108–1116, 2010.

Hyunsoo Kim, Gene H Golub, and Haesun Park. Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics*, 21(2): 187–198, 2005.

Ki-Yeol Kim, Byoung-Jin Kim, and Gwan-Su Yi. Reuse of imputed data in microarray analysis increases imputation efficiency. *BMC Bioinformatics*, 5(1):1, 2004.

Alan Wee-Chung Liew, Ngai-Fong Law, and Hong Yan. Missing value imputation for gene expression data: computational techniques to recover missing data from available information. *Briefings in Bioinformatics*, 12(5):498–513, 2011.

Roderick JA Little and Donald B Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 1987.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11(Aug): 2287–2322, 2010.

Shakir Mohamed, Zoubin Ghahramani, and Katherine A Heller. Bayesian exponential family PCA. In *Advances in Neural Information Processing Systems*, pages 1089–1096, 2009.

Shigeyuki Oba, Masa-aki Sato, Ichiro Takemasa, Morito Monden, Ken-ichi Matsubara, and Shin Ishii. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

Trivellore E Raghunathan, James M Lepkowski, John Van Hoewyk, and Peter Solenberger. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, 27(1):85–96, 2001.

Daniel J Stekhoven and Peter Bühlmann. Missforest: non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

Stef Van Buuren. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, 16(3):219–242, 2007.

Kiri Wagstaff. Clustering with missing values: No imputation required. In *Classification, Clustering, and Data Mining Applications*, pages 649–658. Springer, 2004.

Xian Wang, Ao Li, Zhaohui Jiang, and Huanqing Feng. Missing value estimation for DNA microarray gene expression data by support vector regression imputation and orthogonal coding scheme. *BMC Bioinformatics*, 7(1):1, 2006.

Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1): 3–34, 2015.

Xiaobai Zhang, Xiaofeng Song, Huinan Wang, and Huanping Zhang. Sequential local least squares imputation estimating missing value of microarray data. *Computers in Biology and Medicine*, 38(10):1112–1120, 2008.