15.095 Machine Learning with Optimization

Lecture 7: Missing Data Imputations

Overview

Outline

Introduction

Missing data appear in almost all applications:

- Electronic health records
- Genomics
- Time series
- Survey response
- ...

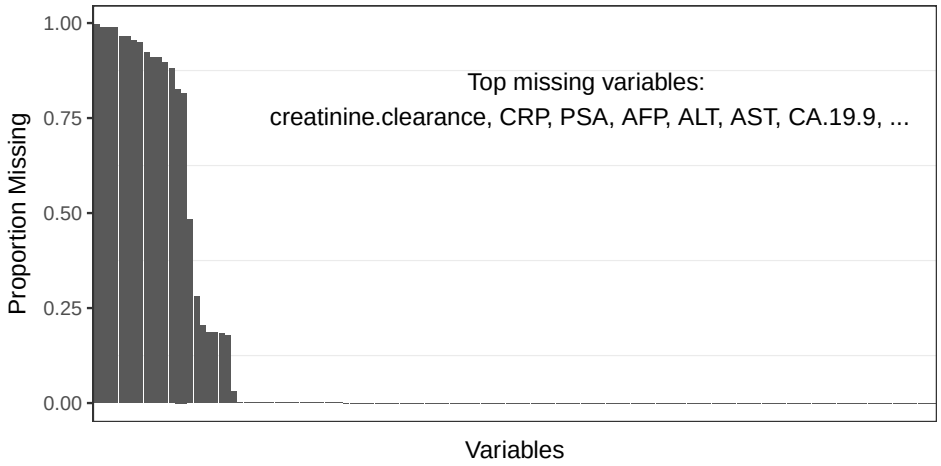


Figure: Example of missing data pattern from a cancer mortality prediction project. More than 20% of variables have missing values, many with more than 50% observations missing.

The Missing Data Problem

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ with some missing values, our goal is to generate a complete data matrix $\mathbf{X}^{imp} \in \mathbb{R}^{n \times p}$ with imputed values $w_{id}, (i,d) \in \mathcal{M}$, where:

- $n =$ number of observations,
- $p =$ number of features,
- $x_{id} =$ the dth feature of observation i,
- $\mathcal{M} = \{(i,d) : x_{id} \text{ is missing}\}$.

$$\mathbf{X} \quad \Rightarrow \quad \mathbf{X}^{imp}$$

$$\begin{bmatrix} x_{11} & x_{12} & ? & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & ? & ? & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \\ x_{51} & x_{52} & ? & x_{54} \\ x_{61} & x_{62} & x_{63} & x_{64} \\ x_{71} & x_{72} & ? & ? \\ x_{81} & ? & x_{83} & x_{84} \end{bmatrix} \Rightarrow \begin{bmatrix} x_{11} & x_{12} & w_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & w_{32} & w_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \\ x_{51} & x_{52} & w_{53} & x_{54} \\ x_{61} & x_{62} & x_{63} & x_{64} \\ x_{71} & x_{72} & w_{73} & w_{74} \\ x_{81} & w_{82} & x_{83} & x_{84} \end{bmatrix}$$

Mechanism of Missing Data

- Missing completely at random (MCAR):

$$f(\mathcal{M}|\mathbf{X}) = f(\mathcal{M})$$

- Missing at random (MAR):

$$f(\mathcal{M}|\mathbf{X}) = f(\mathcal{M}|\mathbf{X}_{\text{obs}})$$

- Not missing at random (NMAR), all other cases
  - E.g., Survival analysis
  - Customer surveys

Introduction

Depending on the application, imputation is useful for:

1. The completed dataset itself (e.g., "the Netflix problem"),
2. Statistical inference,
3. Using on a downstream machine learning model.

Simple solutions to the missing data problem include:

1. **Complete-case analysis**: Ignore all observations (rows) with missing values.
2. **Complete-feature analysis**: Ignore all features (columns) with missing values.
3. **Mean imputation**: Impute each missing value as the row/column mean.

Outline

History

- **Expectation-Maximization (EM)** [Dempster et al., 1977]
- **Predictive-Mean Matching (PMM)** [Little, 1988]
- **K-Nearest Neighbors (KNN)** [Troyanskaya et al., 2001]
  - Sequential KNN [Kim et al., 2004]
  - Iterative KNN [Brás and Menezes, 2007]
- **Singular Value Decomposition (SVD)** [Troyanskaya et al., 2001]
  - Bayesian PCA [Oba et al., 2003]
  - Soft-threshold SVD [Mazumder et al., 2010]
- **Least Squares (LS)** [Raghunathan et al., 2001, Bø et al., 2004]
  - Local-Least Squares (LLS) [Kim et al., 2005]
  - Sequential LLS [Zhang et al., 2008]
- **Classification and Regression Trees (CART)** [Burgette and Reiter, 2010]
  - Random Forest [Stekhoven and Bühlmann, 2012]
- **Support Vector Machines (SVM)** [Wang et al., 2006]

Example: EM imputation

- Maximimum Likelihood Estimator (MLE):

$$\max \; L(\mathbf{X}_{\text{obs}}|\theta)$$

- A heuristic iterative approach:
  1. Expectation step (expected complete data likelihood):

$$q(\theta|\theta^*) \leftarrow \mathbb{E}[L(\theta|\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{miss}})|\mathbf{X}_{\text{obs}}, \theta^*)]$$

  2. Maximization step (maximize θ):

$$\theta^* \leftarrow \arg\max_{\theta} \; q(\theta|\theta^*)$$

  3. Iterate until convergence.
- Example: Gaussian or mixture of Gaussian models.
- Issues: **strong distributional assumptions**, breaks with singular covariance matrices

Example: Multiple Imputation with PMM

- Multiple imputation: useful for statistical inference.
- Idea: to capture the variance of imputed missing values by pooling many different imputed results.
- Predictive-Mean Matching (PMM): a heuristic method that fits a linear model predicting one column and randomly choosing from the posterior.
- Generalize when multiple columns have missing values (MICE package).
- Issues: **slow**, **not scale** to high dimensions, **not very accurate**

Example: KNN Impute

- Impute each missing value $x_{id}, (i,d) \in \mathcal{M}$ based upon the K points closest to $\mathbf{x}_i$
  - Only points with known values in dimension d are used as potential neighbors.
  - When calculating the distance, if a dimension r is missing, that dimension is not used.
- Widely used for imputation of microarray data sets.
- Issues: highly heuristic, **performs poorly with large missing percentages**, as many data points cannot be used as potential neighbors.
- Some extensions such as sequential or iterative KNN were developed, but mostly heuristic fixes on a heuristic method, no significant improvement in performance

Outline

An Optimization Perspective

1. Provides a general modeling framework for the missing data problem.
2. Does not require strong assumptions about the underlying distribution of missing data.
3. Scales to large problem sizes ($n \sim$ 100,000's, $p \sim$ 1,000's).
4. Produces imputations competitive with existing methods, especially as the % of missing data increases.

opt.knn

Objective

Minimize the sum of Euclidean distances from each imputed value $\mathbf{w}_i$ to its K nearest neighbors.

Decision variables:

$$w_{id} \qquad (i,d) \in \mathcal{M}.$$

Known values:

$$w_{id} = x_{id} \qquad (i,d) \notin \mathcal{M}.$$

Auxiliary variables:

$$z_{ij} = \begin{cases} 1, & \text{if } \mathbf{w}_j \text{ is among the } K\text{-nearest neighbors of } \mathbf{w}_i, \\ 0, & \text{otherwise.} \end{cases}$$

opt.knn

$$\begin{aligned}
\min \quad & c(\mathbf{Z}, \mathbf{W}, \mathbf{X}) := \sum_{i \in \mathcal{I}} \sum_{j \neq i} z_{ij} \|\mathbf{w}_i - \mathbf{w}_j\|_2^2, \\
\text{s.t.} \quad & w_{id} = x_{id}, && (i,d) \notin \mathcal{M}, \\
& \sum_{j \neq i} z_{ij} = K, && i \in \mathcal{I}, \\
& \mathbf{Z} \in \{0,1\}^{|\mathcal{I}| \times (n-1)},
\end{aligned}$$

where

$$\mathcal{I} = \{i : \mathbf{x}_i \text{ has at least one missing coordinate}\}.$$

opt.knn

Goal: Solve this problem for large data sets with $n \sim$ 100,000's, $p \sim$ 1,000's.

- This is a non-convex, nonlinear integer optimization problem.
- Global optimal solution methods not scale.
- Thus, we use fast first-order methods with random warm starts:
  - Block Coordinate Descent (BCD)
  - Coordinate Descent (CD)
- We find high-quality solutions for large data sets in minutes.

Block Coordinate Descent for K-NN

Given a warm-start $\mathbf{W}$ of imputed values, we alternate updating $(\mathbf{Z}, \mathbf{W})$ until these values converge to a local minimum of our global optimization problem:

$$\begin{aligned}
\min \quad & c(\mathbf{Z}, \mathbf{W}; \mathbf{X}) := \sum_{i \in \mathcal{I}} \sum_{j \neq i} z_{ij} \|\mathbf{w}_i - \mathbf{w}_j\|_2^2, & \\
\text{s.t.} \quad & w_{id} = x_{id}, & (i,d) \notin \mathcal{M}, \\
& \sum_{j \neq i} z_{ij} = K, & i \in \mathcal{I}, \\
& \mathbf{Z} \in \{0,1\}^{|\mathcal{I}| \times (n-1)}.
\end{aligned}$$

Block Coordinate Descent for K-NN

Fixing $\mathbf{W}$, the update for $\mathbf{Z}$ is a simple sorting procedure.

Taking the derivative w.r.t. z_{ij}, for some $i \in \mathcal{I}$, we obtain:

$$\frac{\partial c(\mathbf{Z}, \mathbf{W}; \mathbf{X})}{\partial z_{ij}} = \|\mathbf{w}_i - \mathbf{w}_j\|_2^2.$$

Since we also have the constraint

$$\sum_{j \neq i} z_{ij} = K \qquad \forall i \in \mathcal{I},$$

the optimal solution is:

$$z_{ij} = \begin{cases} 1, & \text{if } \mathbf{w}_j \text{ is among the } K\text{-nearest neighbors of } \mathbf{w}_i, \\ 0, & \text{otherwise.} \end{cases}$$

Block Coordinate Descent for K-NN

Fixing $\mathbf{Z}$, the update for $\mathbf{W}$ is the solution to a quadratic problem.

Taking the derivative w.r.t. w_{id}, for some $(i,d) \in \mathcal{M}$, we obtain:

$$\begin{aligned}\frac{\partial c(\mathbf{Z},\mathbf{W};\mathbf{X})}{\partial w_{id}} = \quad & (K + \textstyle\sum_{j\in\mathcal{I}} z_{ji}) w_{id} - \sum_{(j,d)\in\mathcal{M}} (z_{ij} + z_{ji}) w_{jd} \\ & - \textstyle\sum_{(j,d)\notin\mathcal{M}} (z_{ij} + \mathbb{1}_{\{j\in\mathcal{I}\}} z_{ji}) x_{jd}.\end{aligned}$$

For each dimension d, we have a system of equations of the form:

$$\frac{\partial c(\mathbf{Z},\mathbf{W};\mathbf{X})}{\partial w_{id}} = 0, \qquad \forall (i,d) \in \mathcal{I}.$$

We can also represent this linear system in matrix notation as $\mathbf{Q}\mathbf{w}^d = \mathbf{R}\mathbf{x}^d$, where $\mathbf{w}^d$, $\mathbf{x}^d$ are the missing, known values in dimension d. WLOG $\mathbf{Q} \succeq 0$, so there is a closed-form solution $\mathbf{w}^d = \mathbf{Q}^{-1}\mathbf{R}\mathbf{x}^d$ for each d.

Block Coordinate Descent for K-NN

Supposing that entries $1, \ldots, a$ are unknown and $(a+1), \ldots, n$ are known in dimension d, we have:

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

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

Coordinate Descent for K-NN

Alternatively, given a warm-start $\mathbf{W}$ of imputed values, in **Coordinate Descent** we update each coordinate w_{id} until these values converge to a local minimum of our global optimization problem:

$$\begin{aligned} \min \quad & c(\mathbf{Z}, \mathbf{W}; \mathbf{X}) := \sum_{i \in \mathcal{I}} \sum_{j \neq i} z_{ij} \|\mathbf{w}_i - \mathbf{w}_j\|_2^2, \\ \text{s.t.} \quad & w_{id} = x_{id}, && (i,d) \notin \mathcal{M}, \\ & \sum_{j \neq i} z_{ij} = K, && i \in \mathcal{I}, \\ & \mathbf{Z} \in \{0,1\}^{|\mathcal{I}| \times (n-1)}. \end{aligned}$$

The update for $\mathbf{Z}$ is identical to the update in BCD.

Coordinate Descent for K-NN

For each $(i, d) \in \mathcal{M}$, we solve the following quadratic problem:

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

where $x_{jd} := w_{jd} \quad \forall j \neq i$. This results in the update:

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

This can be interpreted as a weighted average of the K nearest neighbors of $\mathbf{x}_i$, along with all points $\mathbf{x}_j$ that include $\mathbf{x}_i$ as a neighbor.

Outline

General Optimization Formulation

We can extend this approach to handle categorical variables and also to use other imputation models, such as decision trees and SVM. For each model:

- Imputed continuous variables: $w_{id}, (i,d) \in \mathcal{M}_0$,
- Imputed categorical variables: $v_{id}, (i,d) \in \mathcal{M}_1$,
- Known continuous variables: $w_{id}, (i,d) \in \mathcal{N}_0$,
- Known categorical variables: $v_{id}, (i,d) \in \mathcal{N}_1$,
- Auxiliary variables: $\mathbf{U}$ (e.g. z_{ij} for K-NN),
- Cost function: $c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X})$,
- Constraints: $(\mathbf{U}, \mathbf{W}, \mathbf{V}) \in \mathcal{U}$.

General Optimization Formulation

We obtain the following optimization problem:

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

where $\mathcal{U}$ is the set of all feasible combinations $(\mathbf{U}, \mathbf{W}, \mathbf{V})$ of auxiliary variables and imputed values.

opt.knn

Suppose that features $1, \ldots, p_0$ are continuous, and features $(p_0 + 1), \ldots, (p_0 + p_1)$ are categorical.

$$\min \quad c(\mathbf{Z}, \mathbf{W}, \mathbf{V}; \mathbf{X}) := \sum_{i \in \mathcal{I}} \sum_{j=1}^{n} z_{ij} \left[\sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} \mathbb{1}_{\{v_{id} \neq v_{jd}\}} \right]$$

$$\begin{aligned}
\text{s.t.} \quad & w_{id} = x_{id} && (i,d) \in \mathcal{N}_0, \\
& v_{id} = x_{id} && (i,d) \in \mathcal{N}_1, \\
& z_{ii} = 0 && i \in \mathcal{I}, \\
& \sum_{j=1}^{n} z_{ij} = K && i \in \mathcal{I}, \\
& \mathbf{Z} \in \{0,1\}^{|\mathcal{I}| \times n}
\end{aligned}$$

where

$$\mathcal{I} = \{i : \mathbf{x}_i \text{ has at least one missing coordinate}\}.$$

opt.svm

Suppose that features $1, \ldots, p_0$ are continuous, and features $(p_0+1), \ldots, (p_0+p_2)$ are discrete $\in \{-1, 1\}$.

For continuous features, we use **SVM for regression**:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\beta\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n}\sum_{d=1}^{p_0}(\gamma_{id}+\gamma_{id}^*) \\
\text{s.t.} \quad & w_{id} = x_{id}, && (i,d) \in \mathcal{N}_0, \\
& \beta_{dd} = 0 && d = 1, \ldots, p_0, \\
& \gamma_{id} \geq w_{id} - (\beta_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - \epsilon && d = 1, \ldots, p_0, i = 1 \ldots, n, \\
& \gamma_{id}^* \geq (\beta_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon && d = 1, \ldots, p_0, i = 1 \ldots, n, \\
& \gamma_{id} \geq 0 && d = 1, \ldots, p_0, \\
& \gamma_{id}^* \geq 0 && d = 1, \ldots, p_0.
\end{aligned}
$$

opt.svm

Suppose that features $1, \ldots, p_0$ are continuous, and features $(p_0+1), \ldots, (p_0+p_2)$ are discrete $\in \{-1, 1\}$.

For discrete features, we use **SVM for classification**:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{\theta}\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n}\sum_{d=p_0+1}^{p_0+p_2}\xi_{id} \\
\text{s.t.} \quad & \widetilde{v}_{id} = x_{id}, && (i,d) \in \mathcal{N}_2, \\
& \theta_{dd} = 0 && d = (p_0+1), \ldots, (p_0+p_2), \\
& \xi_{id} \geq 1 - \widetilde{v}_{id}(\boldsymbol{\theta}_d^T \begin{bmatrix} \mathbf{w}_i \\ \widetilde{\mathbf{v}}_i \end{bmatrix} + \theta_{d0}) && d = (p_0+1), \ldots, (p_0+p_2), i = 1 \ldots, n, \\
& \xi_{id} \geq 0 && d = (p_0+1), \ldots, (p_0+p_2), i = 1 \ldots, n, \\
& \widetilde{v}_{id} \in \{-1, 1\} && d = (p_0+1), \ldots, (p_0+p_2), i = 1 \ldots, n,
\end{aligned}
$$

where $\widetilde{v}_{id}$ are one-hot encodings of the original categorical variables.

opt.svm

$$\min \quad c([\beta,\theta],\mathbf{W},\widetilde{\mathbf{V}};\mathbf{X}) := \frac{1}{2}\left(\|\theta\|_{\mathcal{H}}^2 + \|\beta\|_{\mathcal{H}}^2\right) + C\left(\sum_{i=1}^{n}\sum_{d=1}^{p_0}(\gamma_{id}+\gamma_{id}^*) + \sum_{i=1}^{n}\sum_{d=p_0+1}^{p_0+p_2}\xi_{id}\right)$$

$$\begin{aligned}
\text{s.t.} \quad & x_{id} = w_{id} && (i,d) \in \mathcal{N}_0,\\
& \widetilde{v}_{id} = \widetilde{v}_{id}^{fixed} && (i,d) \in \mathcal{N}_2,\\
& \beta_{dd} = 0 && d = 1,\dots,p_0,\\
& \theta_{dd} = 0 && d = 1,\dots,p_2,\\
& \gamma_{id} \geq w_{id} - (\beta_d^T \begin{bmatrix}\mathbf{w}_i\\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \beta_{d0}) - \epsilon && d = 1,\dots,p_0, i = 1\dots,n,\\
& \gamma_{id}^* \geq (\beta_d^T \begin{bmatrix}\mathbf{w}_i\\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \beta_{d0}) - w_{id} - \epsilon && d = 1,\dots,p_0, i = 1\dots,n,\\
& \xi_{id} \geq 1 - \widetilde{v}_{id}(\theta_d^T \begin{bmatrix}\mathbf{w}_i\\ \widetilde{\mathbf{v}}_i\end{bmatrix} + \theta_{d0}) && d = 1,\dots,p_2, i = 1\dots,n,\\
& \gamma_{id} \geq 0 && d = 1,\dots,p_0, i = 1\dots,n,\\
& \gamma_{id}^* \geq 0 && d = 1,\dots,p_0, i = 1\dots,n,\\
& \xi_{id} \geq 0 && d = 1,\dots,p_2, i = 1\dots,n,\\
& \widetilde{v}_{id} \in \{-1,1\} && d = 1,\dots,p_2, i = 1\dots,n.
\end{aligned}$$

$\mathbf{U} = [\mathbf{T}]$

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

where

$$t_{ij}^d = \begin{cases} 1, & \text{if } (\mathbf{w}_j, \mathbf{v}_j) \text{ is in the same leaf node as } (\mathbf{w}_i, \mathbf{v}_i) \\ & \text{within the decision tree to predict feature } d, \\ 0, & \text{otherwise.} \end{cases}$$

The tree constraints are $(\mathbf{T}^d, \mathbf{W}, \mathbf{V}) \in \mathcal{T}^d$ for each dimension [Bertsimas and Dunn, 2017].

Summary of Optimization Formulations

Model	U	$c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X})$
K-NN	$\mathbf{Z}$	$\sum_{i \in \mathcal{I}} \sum_{j=1}^{n} z_{ij} \left[\sum_{d=1}^{p_0} (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} \mathbb{1}_{\{v_{id} \neq v_{jd}\}} \right]$
SVM	$[\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\xi}]$	$\frac{1}{2}(\|\boldsymbol{\beta}\|_{\mathcal{H}}^2 + \|\boldsymbol{\theta}\|_{\mathcal{H}}^2) + C \sum_{i=1}^{n} \left(\sum_{d=1}^{p_0} (\gamma_{id} + \gamma_{id}^*) + \sum_{d=p_0+1}^{p_0+p_2} \xi_{id} \right)$
Trees	$\mathbf{T}$	$\sum_{i=1}^{n} \sum_{j=1}^{n} \left[\sum_{d=1}^{p_0} t_{ij}^d (w_{id} - w_{jd})^2 + \sum_{d=p_0+1}^{p_0+p_1} t_{ij}^d \mathbb{1}_{\{v_{id} \neq v_{jd}\}} \right]$

Table: Variables and cost functions for each imputation model.

General `opt.impute` algorithm

Algorithm 1

Initialize $\delta_0 > 0$ and warm start $\mathbf{W}$, $\mathbf{V}$.
In addition, set $\delta \leftarrow \infty$ and $\mathbf{W}^{old} \leftarrow \mathbf{W}$, $\mathbf{V}^{old} \leftarrow \mathbf{V}$.

While $\delta > \delta_0$:

① Update $\mathbf{U}$, the model auxiliary variables:

$$\mathbf{U} \leftarrow \arg\min_{\mathbf{U}} \; c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X}) \quad \text{s.t.} \;\; (\mathbf{U}, \mathbf{W}, \mathbf{V}) \in \mathcal{U}.$$

② Update $\mathbf{W}$ and $\mathbf{V}$, the imputed values, using block coordinate descent (BCD) or coordinate descent (CD).

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

Return $\mathbf{X}^{imp} \leftarrow \mathbf{W}, \mathbf{V}$.

Convergence Properties

- $c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X})$ is non-increasing under both BCD and CD updates.
- Algorithm 1 is guaranteed to terminate within a finite number $(\lceil \frac{1}{\delta_0} c(\mathbf{U}^{init}, \mathbf{W}^{init}, \mathbf{V}^{init}; \mathbf{X}) \rceil)$ of iterations.
- $c(\mathbf{U}, \mathbf{W}, \mathbf{V}; \mathbf{X})$ is non-convex, so convergence to the global optimum is not guaranteed. Using 5 random restarts and selecting the imputation with the lowest obj. value performs well in practice.

Outline

Experimental Setup

- **84 data sets** from the UCI ML Repository, ranging in size from $n = 23$ to 5,875, $p = 2$ to 124.
- Missing percentage ranges from 10% to 50%.
- Missing data generation mechanism: MCAR and NMAR.
- Given the scenario, we generate multiple instances of missing values: $\mathcal{M} = \{(i, d) : x_{id} \text{ is missing}\}$.
- We evaluate downstream tasks for 10 data sets.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \\ x_{51} & x_{52} & x_{53} & x_{54} \\ x_{61} & x_{62} & x_{63} & x_{64} \\ x_{71} & x_{72} & x_{73} & x_{74} \\ x_{81} & x_{82} & x_{83} & x_{84} \end{bmatrix}$$

Figure: Missing data instance with 25% missing.

Experimental Setup

- **84 data sets** from the UCI ML Repository, ranging in size from $n = 23$ to 5,875, $p = 2$ to 124.
- Missing percentage ranges from 10% to 50%.
- Missing data generation mechanism: MCAR and NMAR.
- Given the scenario, we generate multiple instances of missing values: $\mathcal{M} = \{(i, d) : x_{id} \text{ is missing}\}$.
- We evaluate downstream tasks for 10 data sets.

$$\begin{bmatrix} x_{11} & x_{12} & ? & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & ? & x_{34} \\ x_{41} & x_{42} & x_{43} & ? \\ x_{51} & x_{52} & ? & x_{54} \\ x_{61} & ? & x_{63} & x_{64} \\ x_{71} & ? & ? & x_{74} \\ ? & x_{82} & x_{83} & x_{84} \end{bmatrix}$$

Figure: Missing data instance with 25% missing.

Imputation Methods

Individual methods

1. Mean imputation, column averages (`mean`)
2. K-Nearest Neighbors (`knn`)
3. Iterative K-Nearest Neighbors (`iknn`)
4. Predictive-Mean Matching (`pmm`)
5. Bayesian PCA (`bpca`)
6. `opt.knn`
7. `opt.tree`
8. `opt.svm`

Imputation Methods

Cross-validated methods

1. Cross-validated Benchmark (`benchmark.cv`), selects the best from `mean`, `knn`, `iknn`, `pmm`, `bpca`.
2. Cross-validated Optimal Impute (`opt.cv`), selects the best from `opt.knn`, `opt.svm`, `opt.tree`.

Multiple Imputation methods (for downstream tasks only)

1. Multivariate Imputation via Chained Equations (`mice`)
2. Optimal Impute for Multiple Imputation (`opt.mi`)

Performance Measures

1. Imputation performance metric - Mean Absolute Error (MAE):

$$\frac{1}{|\mathcal{M}|} \sum_{(i,d)\in\mathcal{M}} |w_{id} - x_{id}|.$$

2. Downstream task performance by imputation methods:
   1. Classification (OptimalTrees and SVM) - out-of-sample accuracy
   2. Regression (Lasso and SVR) - out-of-sample R^2

Solution Progress

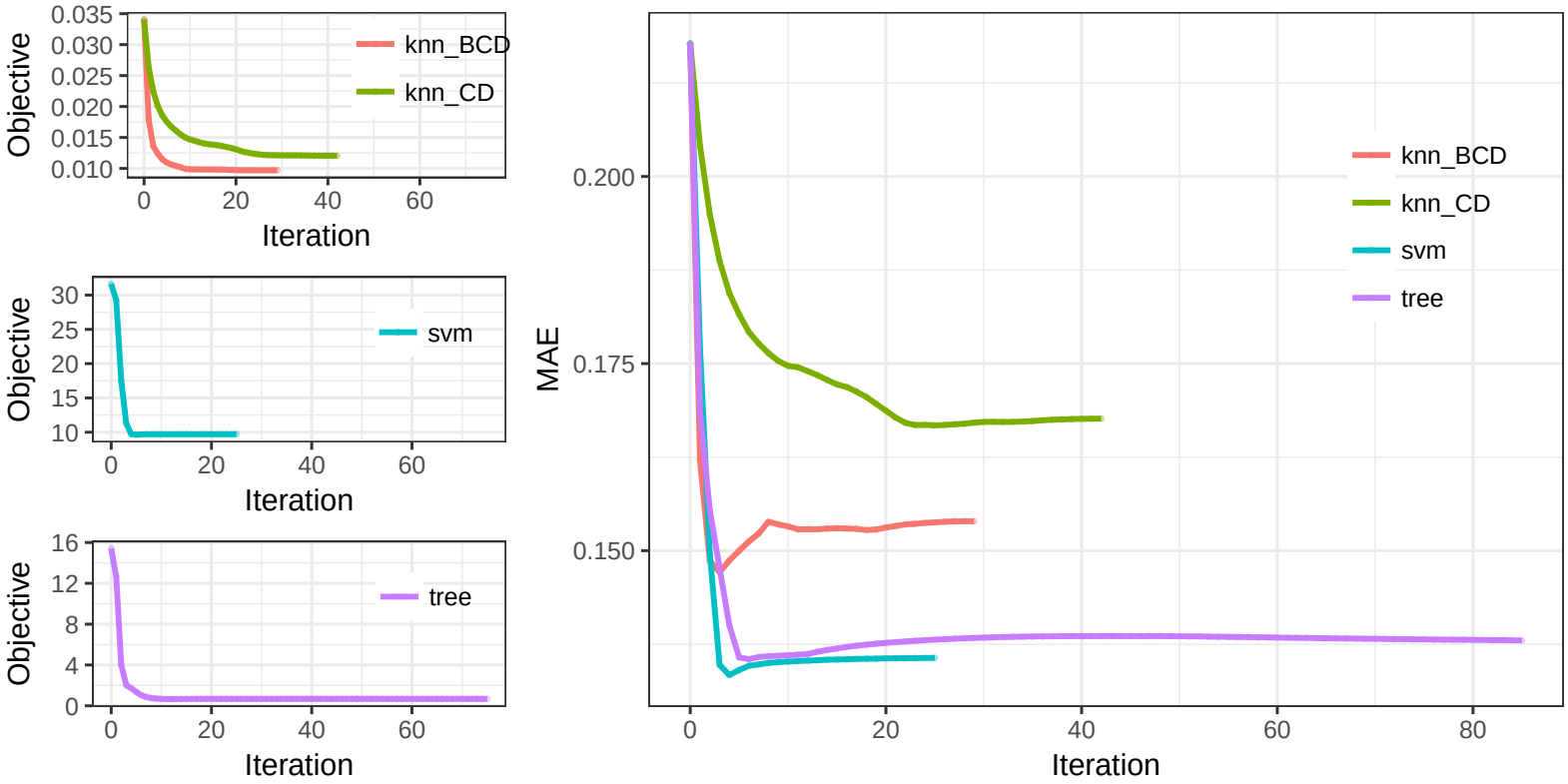


Figure: For each `opt.impute` method over the iterations, convergence is fast (left three plots), and largely leads to better out-of-sample performance (right plot).

Winning Percentage: MAE

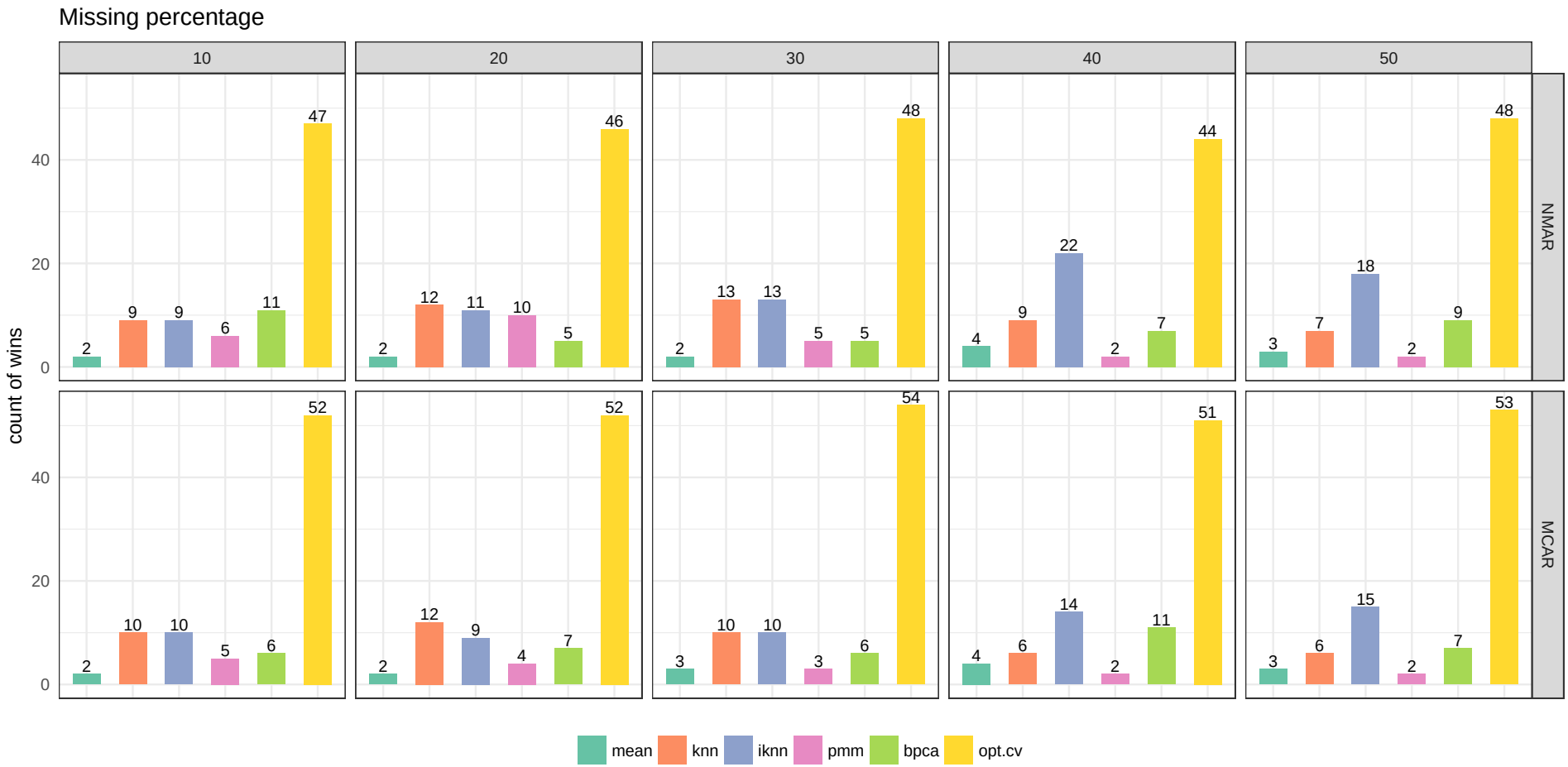


Figure: The best `opt.impute` method (yellow) is selected through cross-validation and compared against other state-of-the-art methods.

Magnitude of Improvement for `opt.impute`

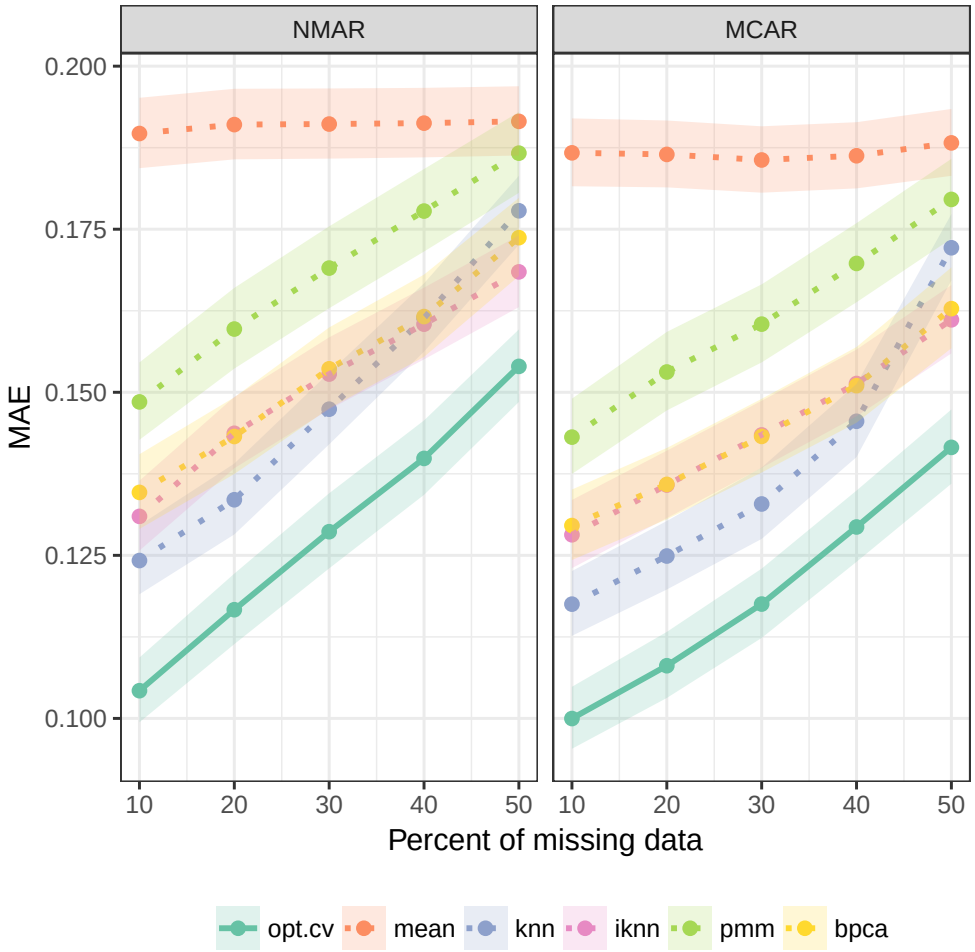


Figure: Average MAE across all data sets for each imputation method, for NMAR and MCAR. The best `opt.impute` method is selected through cross-validation.

Magnitude of Improvement for `opt.impute` over `benchmark.cv`

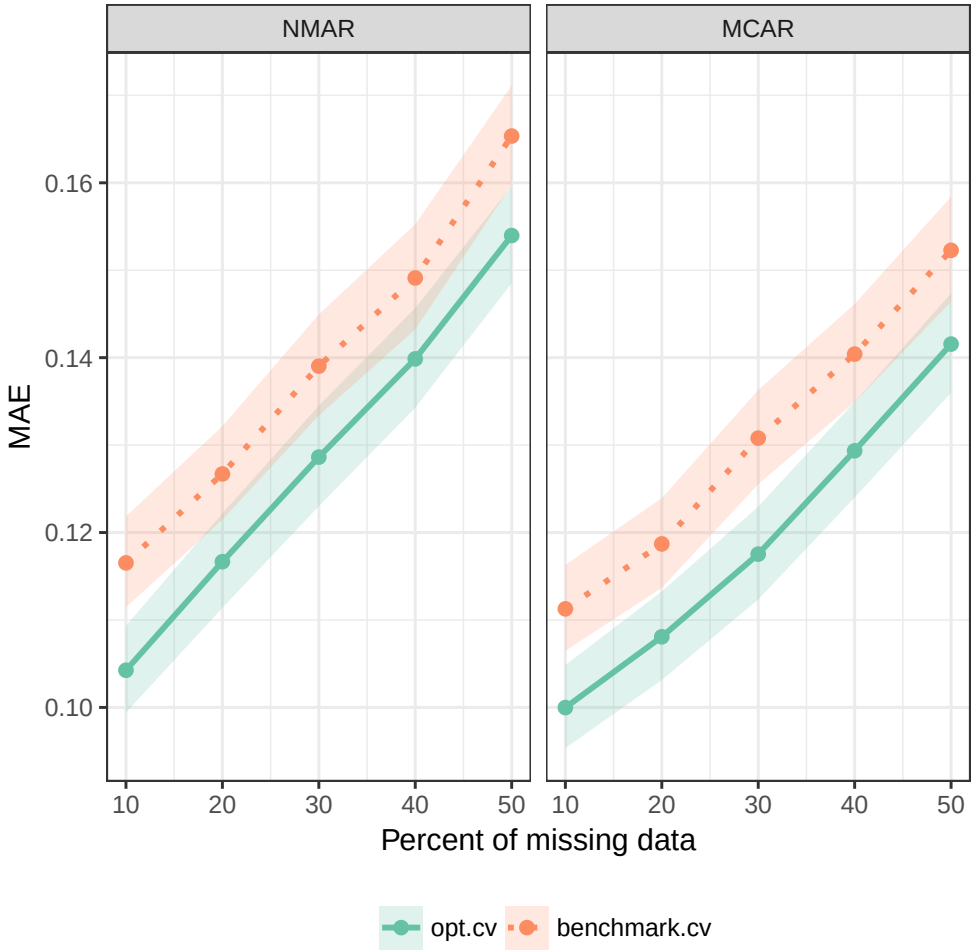
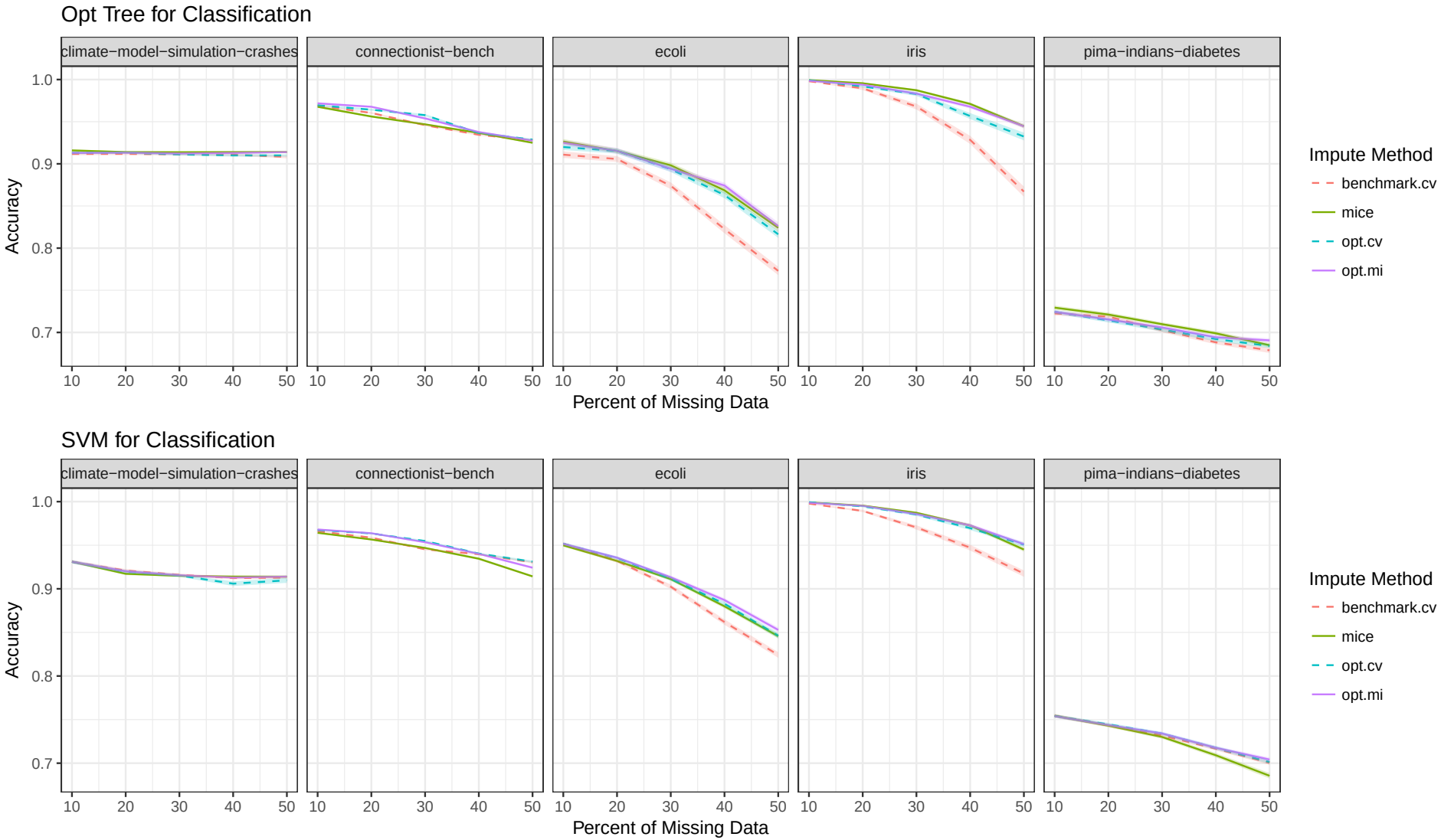
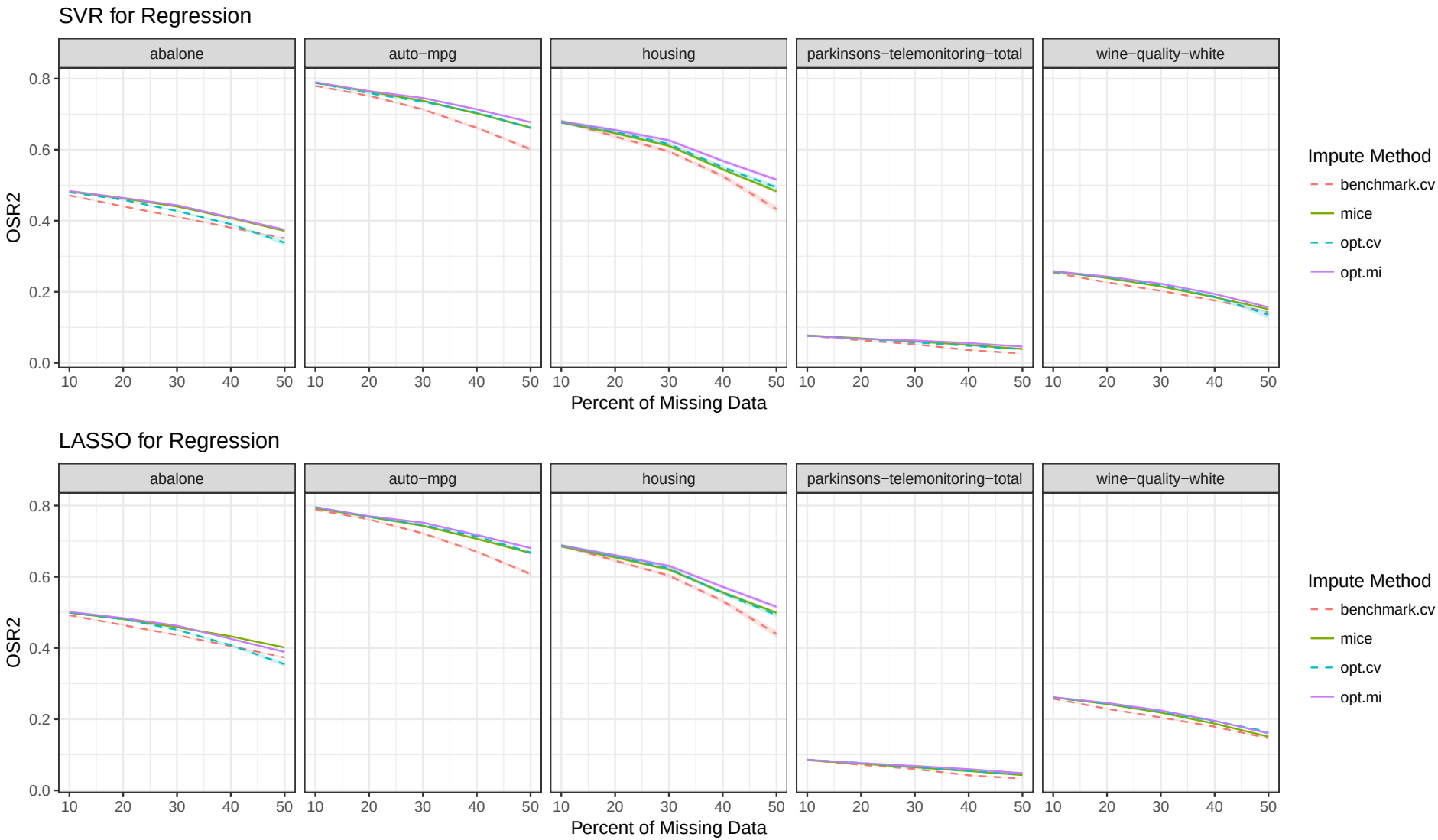


Figure: Average MAE across all data sets for each imputation method, for NMAR and MCAR. The best `opt.impute` method (green) and best benchmark (orange) is selected through cross-validation.

Downstream Task - Classification



Downstream Task - Regression



Downstream Task - Classification

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

Downstream Task - Regression

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

Outline

Conclusions

- Careful treatment of missing data is necessary for statistical inference and applications of many machine learning algorithms.
- Existing methods have short-comings.
- We introduced an optimization-based framework for imputation which is:
  1. **Highly Accurate** compared to existing imputation methods for both imputed values and downstream tasks,
  2. **Scalable** to large data sets with 100,000's of observations,
  3. **Generalizable** to include models beyond K-NN, SVM, and trees.