# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

**5,300**
Open access books available

**130,000**
International authors and editors

**155M**
Downloads

Our authors are among the

**154**
Countries delivered to

**TOP 1%**
most cited scientists

**12.2%**
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

## Interested in publishing with us?
## Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

# Incomplete Data Analysis

*Bo-Wei Chen and Jia-Ching Wang*

## Abstract

This chapter discusses missing-value problems from the perspective of machine learning. Missing values frequently occur during data acquisition. When a dataset contains missing values, nonvectorial data are generated. This subsequently causes a serious problem in pattern recognition models because nonvectorial data need further data wrangling before models are built. In view of such, this chapter reviews the methodologies of related works and examines their empirical effectiveness. At present, a great deal of effort has been devoted in this field, and those works can be roughly divided into two types — Multiple imputation and single imputation, where the latter can be further classified into subcategories. They include deletion, fixed-value replacement, *K*-Nearest Neighbors, regression, tree-based algorithms, and latent component-based approaches. In this chapter, those approaches are introduced and commented. Finally, numerical examples are provided along with recommendations on future development.

**Keywords:** data imputation, missing value analysis, missing data, data wrangling, data analytics

## 1. Introduction

With recent development of the Internet of Things (IoT), communication technology, and wireless sensor networks, a huge amount of data is generated every day [1, 2]. It becomes easier to collect data in quantities than before, but new challenges subsequently arise. For example, in manufacturing industries, manufacturers often utilize and deploy thousands of sensors in production lines to monitor the quality of products and to detect possible anomaly or abnormal events. Those sensing data are stored in databases for further analysis. Nonetheless, the collected data are not always perfect. Missing-value entries may appear in databases. When a dataset contains missing values, it is referred to as an incomplete dataset. Missing values in manufacturing industries frequently occur due to sensor failure, particle occlusion, and physical/chemical interferences [3–5]. Unfortunately, most of them happen due to unknown causes and unforeseen circumstances. In addition to manufacturing industries, missing value problems also frequently occur in biomedical areas, for instance, microarray profiling — One of the commonly used tools in genomics [6]. Microarrays are well known for rapidly automated measurement at massive scale — High-throughput biotechnologies [7], but microarrays suffered from missing-value problems [8]. During microarray profiling, missing values might occur as a result of different reasons, such as human errors, dust or scratches on the slide [9], spotting problems, poor/failed hybridization [10], insufficient image resolutions, and fabrication errors [11]. Those unpredictable factors thereby increase the opportunity of

defect microarrays. In fact, missing-value problems almost challenge every part of our daily applications ranging from manufacturing/biotechnology industries that rely on sensors to typical service industries that involve questionnaire-based surveys. Questionnaires are used to collect information from respondents. Nonetheless, respondents occasionally fail to provide answers that match the format to fit the response categories [12], subsequently generating unanswered questions or invalid formats/responses (e.g., out-of-range values). There are many reasons for such problems, e.g., respondents refused to answer, respondents chose wrong formats [13, 14], respondents intentionally/unintentionally left blanks, testers addressed unclear/confusing choices, designs involved sensitive/private questions, and interviews were interrupted. These factors could result in missing values in questionnaires.

The difficulty of processing incomplete data is that when a dataset contains missing values, the corresponding entries are marked with invalid values. Accordingly, such a dataset becomes nonvectorial because invalid values are present (which are constantly represented as Not-a-Number (NaN)). To tackle those entries with NaN, mathematical operations (e.g., pairwise distance) need further revision under such circumstances because nonvectorial arithmetic is not well defined.

To handle missing-value problems, data imputation is generally used. Data imputation is a statistical term that describes the process of substituting estimated values for missing ones. Related approaches for data imputation [15] can be classified into two types: Multiple imputation and single imputation. The former is aimed at generating two distributions. One is a distribution for selecting hyperparameters, and the other is a distribution for generating data. Multiple imputation uses a function for generating distributional hyperparameters and takes samples from such a function to obtain an averaged distributional hyperparameter set. Multiple imputation then utilizes this averaged distributional hyperparameter set to create a statistical distribution for describing the data distribution. Finally, data samples are drawn to replace missing values. Popular methods for multiple imputation include Expectation Maximization (EM) algorithms or Monte Carlo Markov Chain (MCMC) strategies [16, 17]. Regarding single imputation, it does not involve drawing data samples from an uncertain function to substitute for missing data as multiple imputation does. In brief, single imputation relies on neither sample drawing nor uncertain functions. At present, a great deal of effort has been devoted to single imputation, for example, hot-deck/cold-deck, deletion, fixed-value replacement (e.g., zeros, means, and medians), *K*-Nearest Neighbors (KNNs) [18], regression [10, 19, 20], tree-based algorithms [21, 22], and latent component-based approaches (including matrix completion) [15, 23, 24]. This chapter focuses on *K*-Nearest Neighbors, regression, tree-based algorithms, and latent component-based approaches because the imputation errors of deck/cold-deck and fixed-value replacement are not satisfactory. Moreover, deletion could result in loss of discriminant features or samples. Therefore, the subsequent sections lay emphasis on the other methods.

The rest of this chapter is organized as follows. Section 2 introduces related works, and their methods are subsequently introduced. Sections 3 shows the numerical results, and finally the conclusions are drawn in Section 4.

## 2. Imputation methods

The following subsections introduce data imputation using KNNs, regression, tree-based algorithms, and latent component-based approaches, respectively. For clarity, the description on the dataset uses the following definitions and notations. A nonmissing-value dataset is represented as $\mathbf{X} = \{\mathbf{x}_n \,|n = 1, 2, \ldots, N\}$, where

$\mathbf{x}_n \in \mathbb{R}^M$, $N \in \mathbb{Z}_+$ refers to the number of samples, and the dimensionality of a sample is $M \in \mathbb{Z}_+$. Herein, the dimensionality represents the number of independent variables, predictor variables, features, dimensions, control variables, or explanatory variables. Those terms are used, depending on various research fields, e.g., data science, machine learning, and statistics. Furthermore, $\mathbf{X}$ is max-min normalized.

If supervised learning is required, the label information and the response variable corresponding to each sample $\mathbf{x}_n$ are respectively defined as $c_n$ and $y_n$. The former belongs to categorical variables $\in \mathbb{R}$ after encoded, and $\mathbf{c} = \{c_n | n = 1, 2, \ldots, N\}$. The latter belongs to numerical variables $\in \mathbb{R}$, and $\mathbf{y} = \{y_n | n = 1, 2, \ldots, N\}$. Moreover, the sizes of $\mathbf{X}$, $\mathbf{c}$, and $\mathbf{y}$ are $M$-by-$N$, 1-by-$N$, and 1-by-$N$. For $\mathbf{X}$, $\mathbf{c}$, and $\mathbf{y}$ that contain missing values, $\tilde{\mathbf{X}}$, $\tilde{\mathbf{c}}$, and $\tilde{\mathbf{y}}$ are used.

### 2.1 Imputation based on *K*-nearest neighbors

KNNImpute [9] is a popular imputation tool that leverages KNN algorithms to find the $K$-nearest neighbors nearby a given sample $\tilde{\mathbf{x}}_t$ that contains missing values (if no missing values are present, $\mathbf{x}_t \in \mathbb{R}^M$). The substituted values are generated based on the weighted average of those $K$-nearest neighbors. Notably, there is a limitation for the selected $K$-nearest neighbors when KNNImpute is executed. That is, the dimensions of those $K$-nearest neighbors corresponding to missing-value entries should contain nonmissing-value data. In the plain version, label information was not used while the $K$-nearest neighbors were searched (**Table 1**).

In the above-mentioned algorithm, ":" means selecting the entire rows or columns based on the position, and the operator $\oplus$ means to replace the missing values with the corresponding generated substituted values. Moreover, distance$(\cdot,\cdot)$ signifies the distance between two samples, e.g., Euclidean or Manhattan distance. Those substituted values were fixed and unchanged once they were generated. Nonetheless, such substituted values were highly affected by initial conditions, such as the subset of $M$ independent variables and the number of nearest neighbors. Iterative $K$-Nearest Neighbor imputation (IKNNimpute) [25] improved one of such drawbacks by using a loop that iteratively produced substituted values, chose the subset of $M$ independent variables, and reselected near neighbors. **Table 2** lists a simple version of IKNNimpute, where $\tilde{\mathbf{X}}'[j]$ represents the matrix, of which the missing-value entries are filled in with substituted values in the $j$-th iteration, and $J$ denotes the number of iterations. Besides, $\tilde{\mathbf{X}}'$ is formed by horizontally concatenating $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}_t$. Gray KNNs [26] further proposed Gray Relational Analysis to capture

| **Algorithm: KNNImpute** | |
|---|---|
| | **Input**: $\mathbf{X}$ and $\tilde{\mathbf{x}}_t$ |
| | **Output**: $\hat{\mathbf{x}}_t$ |
| 1 | Select $M'$ dimensions ($M' < M$ and $M' \in \mathbb{Z}_+$) without missing values from $\tilde{\mathbf{x}}_t$ |
| 2 | Store the indices of the selected dimensions in an $M'$-by-1 vector $\mathbf{s}$ |
| 3 | Apply KNN algorithm to $(\tilde{\mathbf{x}}_t)_{\mathbf{s},:}$ based on the dataset $\mathbf{X}_{\mathbf{s},:}$ |
| 4 | Store the $K$-nearest samples in an $M'$-by-$K$ matrix $\mathbb{K}$ |
| 5 | Compute a $K$-by-1 weight vector $\mathbf{\Omega} = [1/\text{distance}(k, \text{t}) \vert k = 1, \ldots, K]$ |
| 6 | $(\hat{\mathbf{x}}_t)_{\mathbf{s},:} = (\tilde{\mathbf{x}}_t)_{\mathbf{s},:} \oplus (\mathbb{K}\mathbf{\Omega})$ |

**Table 1.**
*KNNImpute.*

**Algorithm: IKNNImpute**

**Input**: $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{x}}_t$

**Output**: $\hat{\mathbf{X}}$ and $\hat{\mathbf{x}}_t$

Form $\tilde{\mathbf{X}}'$ by horizontally concatenating $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{x}}_t$

Form $\tilde{\mathbf{X}}'[1]$ by filling in the missing-value entries of $\tilde{\mathbf{X}}'$ with initial replacement

**For** $j$ = 1:$J$

  **For** $n$ = 1:($N$ + 1)

    Apply KNN algorithm to $\tilde{\mathbf{x}}_n[j]$ based on the dataset $\tilde{\mathbf{X}}'[j]$

    Store the $K$-nearest samples in an $M$-by-$K$ matrix $\mathbb{K}$

    Compare a $K$-by-1 weight vector $\boldsymbol{\Omega} = [1/\text{distance}(k,n)|k = 1, \ldots, K]$

    $\hat{\mathbf{x}}_n[j+1] = \hat{\mathbf{x}}_n[j] \oplus (\mathbb{K}\boldsymbol{\Omega})$

  **End**

**End**

**Table 2.**
*IKNNImpute.*

pairwise distance between samples, so that near neighbors were appropriately measured and described.

## 2.2 Imputation based on regression

The underlying model of this category is primarily based on well-known Ordinary Least Squares (OLS), which focuses on minimizing least squares errors

$$E_{\text{OLS}} = \left\|\mathbf{y} - \hat{\mathbf{y}}\right\|_2^2 = \left\|\mathbf{y} - \mathbf{w}^\top \mathbf{X}\right\|_2^2 = \text{Tr}\left(\left(\mathbf{y} - \mathbf{w}^\top \mathbf{X}\right)^\top \left(\mathbf{y} - \mathbf{w}^\top \mathbf{X}\right)\right). \tag{1}$$

Herein, $\|\cdot\|_2$ is the $\mathcal{L}_2$-norm distance, $\mathbf{w} \in \mathbb{R}^M$ means an unknown weight vector, $\top$ represents matrix transpose, and $\hat{\mathbf{y}} = \mathbf{w}^\top \mathbf{X}$. Moreover, $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$ is the closed form for finding the weight vector. Given a nonmissing-value sample $\mathbf{x}_t$, of which the response variable $\tilde{y}_t$ is unknown, $\mathbf{w}^\top \mathbf{x}_t$ can generate estimated results. Such methods included Least Squares Imputation (LSImpute) [10], Local LSImpute (LLSImpute) [19], Sequential LLSimpute (SLLSimpute) [27], Iterated LLSImpute (ILLSImpute) [20], Weighted ILLSImpute [28], Regularized LLSImpute (RLLSImpute) [29], and so on.

In LSImpute [10], the authors examined two types of correlations — those among independent variables (i.e., estimating $\tilde{y}_t$ based on $\mathbf{X}$ and $\mathbf{y}$) and those among samples (i.e., estimating $\tilde{y}_t$ based only on the subsets of $\mathbf{y}$). In the latter, $\tilde{y}_t$ and the selected subsets from $\mathbf{y}$ should be highly correlated during OLS modeling [30]. A combined model was also derived by taking the weighted average from the estimated $\hat{y}_t$ based on independent variables and that based on samples. Unlike LSImpute, LLSImpute [19] was aimed at the correlation among independent variables, but the response variable became multivariate, namely, $\mathbf{Y} = \{Y_n | n = 1, 2, \ldots, N\}$, where $Y_n \in \mathbb{R}^L$, and $\mathbf{Y}$ was an $L$-by-$N$ matrix. The fitting weight was converted from $\mathbf{w}$ to an $M$-by-$L$ matrix $\mathbf{W}$. For ILLSImpute, different subsets of independent variables in $\mathbf{X}$ were drawn and examined during iterations to estimate an optimal $\hat{y}_t$. SLLSimpute [27] adopted multistage data imputation, where the whole missing-value entries were divided into multiple groups based on the missing rate of each

independent variable. The groups with lower missing rates were filled in with substituted values first. Then, those recovered groups were utilized for estimating the other groups with higher missing rates during data reconstruction. For other methods like Weighted ILLSImpute [28] and Regularized LLSImpute (RLLSImpute) [29], variants of OLS models were utilized to adapt to data imputation, for example, Locally Weighted OLS, $\mathcal{L}_2$-norm regularized OLS (i.e., ridge regression), $\mathcal{L}_1$-norm OLS (e.g., LASSO (Least Absolute Shrinkage and Selection Operator), Group LASSO, and Sparse Group LASSO), and regression based on other norms [31].

### 2.3 Imputation based on tree-based algorithms

Since Random Forests (RFs) [32] were proposed, the performance of tree-structured classification and regression algorithms were significantly enhanced. Random Forests adopted Bagging (i.e., Bootstrap Aggregating) techniques to perform data sampling and perceptron learning. Multiple trees were created based on randomly selected features several times (where the numbers of features should be smaller than $M$), and each set of sampled data after Bootstrapping was used to train only one tree. A distinct structure used in Random Forests was called proximity matrix, which recorded the concurrence of pairwise samples in a leaf node, namely, the frequencies when two samples coexist in the same leaf node. Such a proximity matrix was symmetric, of which the size was $N$-by-$N$, and it was used as imputation weighting during data recovery.

Based on [22], two basic principles for handling incomplete data can be derived by summarizing the strategies mentioned in [22] — Preimputation [33] and on-the-fly imputation [21]. The former filled in missing-value entries with fixed replacement at the beginning. Replacement was iteratively updated using proximity matrices. RF growing process repeated until convergence. For the latter method, on-the-fly imputation did not fill in missing-value entries with initial substituted values, and it skipped using missing-value samples for computing splitting nodes. When missing-value samples reached leaves, imputation started. As assignment of missing-value samples to leaves involved randomness, iterations until convergence improved imputation.

### 2.4 Imputation based on latent component-based approaches

This type of method has a general procedure for reconstructing an incomplete data matrix. Firstly, the missing-value entries of a data matrix $\tilde{\mathbf{X}}$ are filled in with replacement (e.g., zeros). Secondly, new matrix factors or vector factors are initialized by generating random numbers. In typical, two or three matrix/vector factors are used, e.g., $\mathbf{P} \in \mathbb{R}^{M \times D}$ and $\mathbf{T} \in \mathbb{R}^{D \times N}$. Besides, the product of those factors should be close to $\tilde{\mathbf{X}}$. Thirdly, iterations are performed to improve the replacement. Unlike the aforementioned types of methods, this type has a unique characteristic — Setting the number of latent components $D \in \mathbb{Z}_+$ — before imputation starts.

Popular methods included SVDImpute (i.e., imputation based on Singular Value Decomposition) [34], Nonlinear Iterative Partial Least Squares-Principal Component Analysis (NIPALS-PCA) [35], matrix completion/approximation, and so forth. The common place of SVDImpute and NIPALS-PCA was that projection matrices (or eigenvectors) were computed, whereas plain NMF did not.

#### *2.4.1 SVDImpute*

For SVDImpute, let $\hat{\mathbf{X}}[\iota]$ represent the data matrix, of which the missing-value entries are filled in with substituted values in the $\iota$-th iteration. Subsequently, SVD is perform on $\hat{\mathbf{X}}[\iota]$, so that

$$\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}\right)[\iota] = \mathrm{SVD}\left(\hat{\mathbf{X}}[\iota]\right) \tag{2}$$

where $\mathbf{U}$ is an $M$-by-$M$ unitary matrix, $\mathbf{\Sigma}$ represents an $M$-by-$N$ diagonal matrix, of which the diagonal terms are nonnegative real numbers sorted in descending order, and $\mathbf{V}$ denotes an $N$-by-$N$ matrix. By selecting the top $D$ largest diagonal values from $\mathbf{\Sigma}$, the following two corresponding matrices $\mathbf{P}$ and $\mathbf{T}$ are formed

$$\begin{cases} \mathbf{P}[\iota] = \mathbf{U}_{:,1:D}[\iota] \\ \mathbf{T}[\iota] = \left(\mathbf{\Sigma}\mathbf{V}^{\top}\right)_{1:D,:}[\iota] \end{cases}, \tag{3}$$

where ":,1:$D$" means selecting columns ranging from the first one to the $D$-th one, and "1:$D$,:" extracts rows. The process of (3) is the same as Truncated SVD. Subsequently, the reconstructed data matrix becomes

$$\hat{\mathbf{X}}[\iota+1] = \tilde{\mathbf{X}} \oplus (\mathbf{P}[\iota]\mathbf{T}[\iota]). \tag{4}$$

Herein, the operator $\oplus$ means to replace the missing values of $\tilde{\mathbf{X}}$ with the corresponding generated values by $\mathbf{P}[\iota]\mathbf{T}[\iota]$. Eqs. (2)–(4) iterate until convergence.

### 2.4.2 NIPALS-PCA

As for NIPALS-PCA (abbreviated as NP below), it minimizes the reconstruction error of

$$\begin{aligned} E_{\mathrm{NP}} &= \sum_{m=1}^{M}\sum_{n=1}^{N}\left(\mathbf{H}\odot\left(\hat{\mathbf{X}}-\mathbf{P}\mathbf{T}\right)\right)_{mn}^{2} \\ &= \sum_{n=1}^{N}\left\{\left(\mathbf{H}_{:,n}\odot\left(\hat{\mathbf{X}}_{:,n}-(\mathbf{P}\mathbf{T})_{:,n}\right)\right)^{\top}\left(\mathbf{H}_{:,n}\odot\left(\hat{\mathbf{X}}_{:,n}-(\mathbf{P}\mathbf{T})_{:,n}\right)\right)\right\} \end{aligned}, \tag{5}$$

where $\odot$ is the elementwise multiplication, $\mathbf{H}$ denotes an $M$-by-$N$ index matrix of the nonmissing-value entries in $\tilde{\mathbf{X}}$. That is, if the entry $\tilde{\mathbf{X}}_{m,n}$ is nonmissing, then $\mathbf{H}_{m,n}$ is one. Otherwise, it shows a zero. Let $\mathbf{\Phi} = \mathrm{diag}(\mathbf{H}_{:,n})$ and $\mathbf{\Psi} = \mathrm{diag}(\mathbf{H}_{m,:})$, where $n$ = 1,2, ... ,$N$, and $m$ = 1,2, ... ,$M$. Eq. (5) becomes convex if either $\mathbf{P}$ or $\mathbf{T}$ is fixed. Then, a solution can be achieved based on Alternating Least Squares (ALS). Taking the derivative form of (5) with respect to $\mathbf{T}_{:,n}$ and zeroing the result yield

$$\mathbf{T}_{:,n} = \left(\mathbf{P}^{\top}\mathbf{\Phi}\mathbf{P}\right)^{-1}\mathbf{P}^{\top}\mathbf{\Phi}\hat{\mathbf{X}}_{:,n}. \tag{6}$$

Likewise, taking the derivative form of (5) with respect to $\mathbf{P}_{m,:}$ and arranging the result generate

$$\mathbf{P}_{m,:} = \hat{\mathbf{X}}_{m,:}\mathbf{\Psi}\mathbf{T}^{\top}\left(\mathbf{T}\mathbf{\Psi}\mathbf{T}^{\top}\right)^{-1}. \tag{7}$$

At the beginning, NIPALS-PCA utilizes the SVD result in the first iteration (see (2)) and extracts the top $D$ components from $\mathbf{U}$ and $\mathbf{V}$ as $\mathbf{P}$ and $\mathbf{T}$ (see (3)). Subsequently, alternating computation between $\mathbf{P}$, $\mathbf{T}$, and $\tilde{\mathbf{X}}\oplus(\mathbf{P}\mathbf{T})$ until convergence generates solutions.

### 2.4.3 NMFImpute

Alternating Least Squares has been widely applied to many models, especially matrix completion/approximation. Nonnegative matrix factorization (NMF) is an

important topic in matrix completion/approximation, and it became a highlight when it was applied to recommendation systems in a Netflix contest [36–38]. At present, NMF has developed multiple variants, including (i) regularization based on $\mathcal{L}_1$-norms, $\mathcal{L}_2$-norms [39], $\mathcal{L}_{2,1}$-norms, nuclear norms, mixed norms, and graphs, (ii) different loss functions like Huber loss, the correntropy induced metric [40, 41], Cauchy functions [42], and Truncated Cauchy functions [43, 44], and (iii) many more, such as projected gradient NMF [45], projective NMF [46, 47], and orthogonal NMF [48]. The following uses the plain version of NMF as an example to elaborate the detail. NMF minimizes the Least Squares error of

$$E_{\text{NMF}} = \left\|\hat{\mathbf{X}} - \mathbf{PT}\right\|_2^2 = \text{Tr}\left\{\left(\hat{\mathbf{X}} - \mathbf{PT}\right)^\top \left(\hat{\mathbf{X}} - \mathbf{PT}\right)\right\}, \tag{8}$$

where $\text{Tr}(\cdot)$ is the trace operator. Eq. (8) is nonconvex and hard to solve. When one variable is fixed, (8) becomes convex. One can use ALS or Coordinate Descend to find the solution. Differentiating (8) with respect to $\mathbf{P}_{m,d}$ and $\mathbf{T}_{d,n}$ (where $d = 1,2,\ldots,D$), respectively, yields

$$\begin{cases} \dfrac{\partial E_{\text{NMF}}}{\partial \mathbf{P}_{m,d}} = 2\left(\mathbf{PTT}^\top - \hat{\mathbf{X}}\mathbf{T}^\top\right)_{m,d} \\ \dfrac{\partial E_{\text{NMF}}}{\partial \mathbf{T}_{d,n}} = 2\left(\mathbf{P}^\top\mathbf{PT} - \mathbf{P}^\top\hat{\mathbf{X}}\right)_{d,n} \end{cases}. \tag{9}$$

The multiplicative update rules for (9) are, respectively,

$$\mathbf{P}_{m,d} = \max\left(\varepsilon, \mathbf{P}_{m,d} \odot \left(\frac{\hat{\mathbf{X}}\mathbf{T}^\top}{\mathbf{PTT}^\top}\right)_{m,d}\right) \tag{10}$$

and

$$\mathbf{T}_{d,n} = \max\left(\varepsilon, \mathbf{T}_{d,n} \odot \left(\frac{\mathbf{P}^\top\hat{\mathbf{X}}}{\mathbf{P}^\top\mathbf{PT}}\right)_{d,n}\right), \tag{11}$$

where $\varepsilon$ is an extremely small positive number, and division is elementwise. Eqs. (10), (11), and (4) iterate until convergence.

## 3. Experimental results

To show the imputation performance of the above-mentioned methods, experiments on open datasets were conducted. The datasets included Abalone (Aba), Scene (SCN), White Wine (WW), and Indian Pines (IP). The number of samples were 4177, 2407, 4898, and 21025, respectively. Furthermore, the dimensionality was, respectively, 561, 294, 11, and 200. Imputation approaches included KNN Regression Imputation (KNRImpute), KNNImpute with $K = 5$, Regression Tree Imputation (RTImpute), Random Forest Imputation (RFImpute), and NIPALS-PCA Imputation (PCAImpute) with only one component. All of them were found in open sources.

To generate missing values for each dataset, this study used a random generator to decide missing-values entries. For KNRImpute, KNNImpute, RTImpute, and RFImpute, they required that missing values should not be uniformly distributed in data. Otherwise, imputation could not be performed. Thus, not every of the independent variables were chosen. Missing-value rates ranged from 3.00% to 9.00%,

with a separation of 2.00%. When a dataset was recovered, the difference between the substituted values and the ground truth was compared. The criteria for examining the quality of imputation included root-mean-squared errors (RMSEs) and coefficients of determination ($R^2$). For coefficients of determination, this study reshaped (i.e., vectorized) a dataset into a vector and then used the following definition to compute $R^2$. Assume that $x_g$ represents an element of a ground-truth dataset ($g = 1, \ldots, MN$), $\hat{x}_g$ denotes the corresponding recovered value, and $\overline{x}_g$ denotes the mean of all the ground-truth values in the same dataset

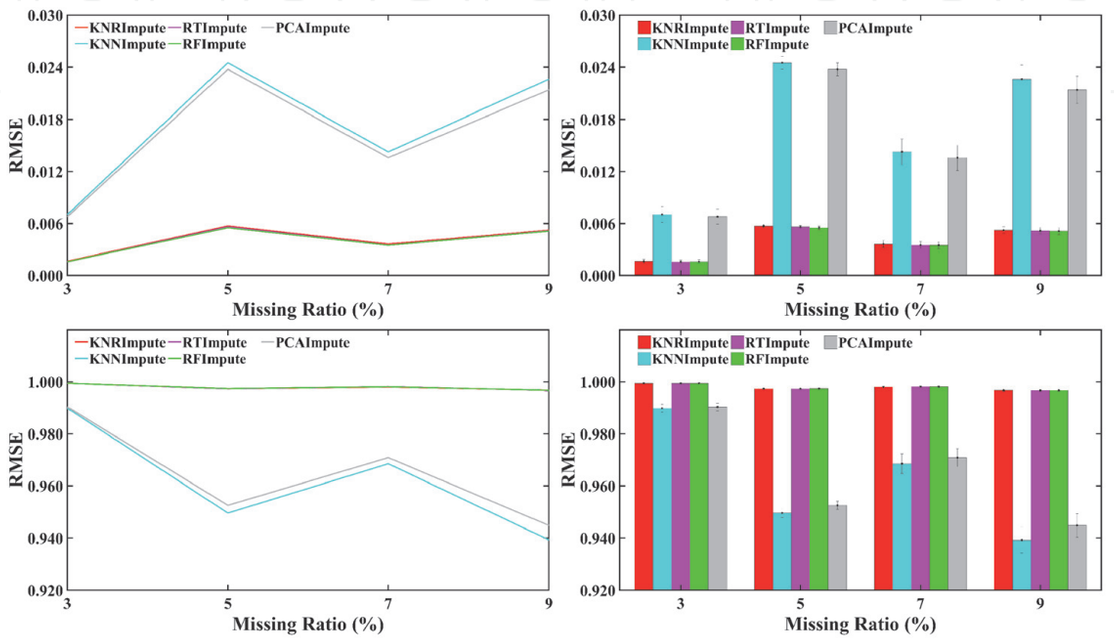


**Figure 1.**
*Average and the standard deviation of the RMSEs and $R^2$ based on dataset Aba. The top ones are RMSEs, and the bottom are $R^2$. The standard deviation was divided by 10.000 for better resolutions.*

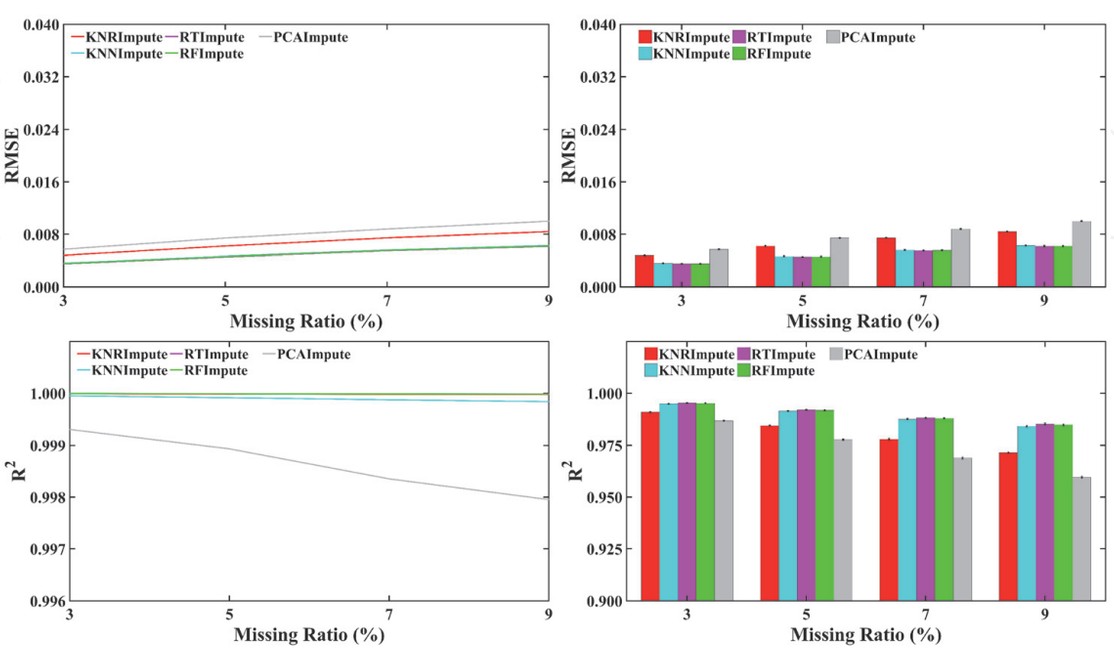


**Figure 2.**
*Average and the standard deviation of the RMSEs and $R^2$ based on dataset SCN. The top ones are RMSEs, and the bottom are $R^2$.*

$$R = 1 - \sum_g \left(x_g - \hat{x}_g\right)^2 / \sum_g \left(x_g - \overline{x}_g\right)^2. \tag{12}$$

When $R$ was close to one, substituted values approached the ground truth. This implied that the difference between the substituted values and the ground truth was smaller.

**Figures 1–4** display the average and the standard deviation of the RMSEs and $R^2$, where the horizontal axis denotes the missing rates, and the vertical axis is the evaluation result. The left subplots are line plots, and the right ones show bar charts with standard variation. As shown in the figures, standard variation was quite small.

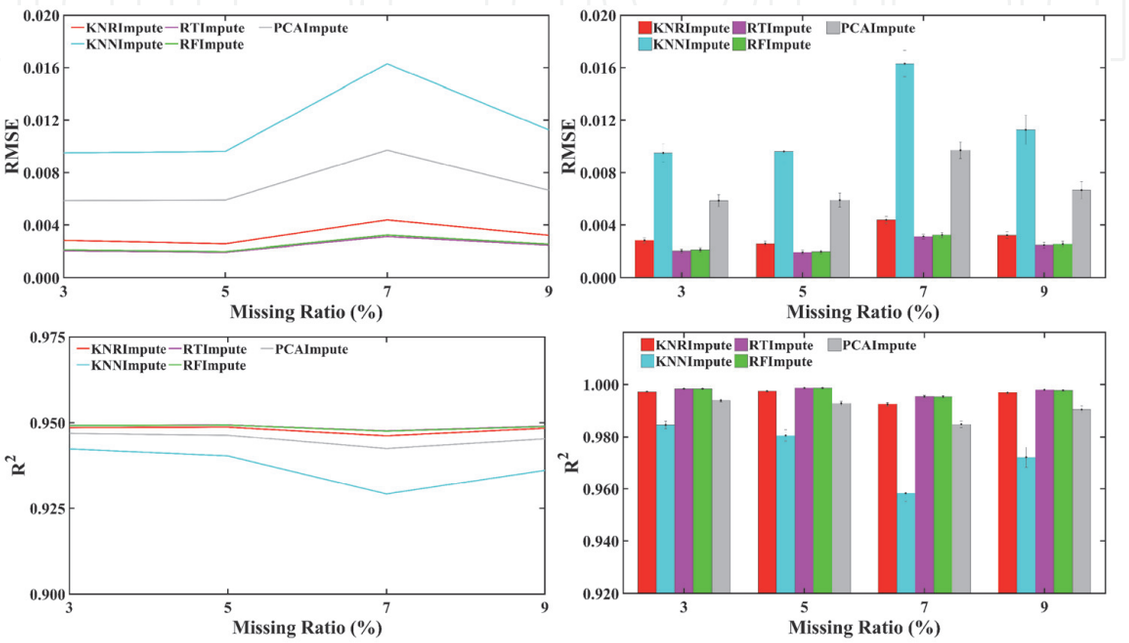


**Figure 3.**
*Average and the standard deviation of the RMSEs and $R^2$ based on dataset WW. The top ones are RMSEs, and the bottom are $R^2$. The standard deviation was divided by 10.000 for better resolutions.*

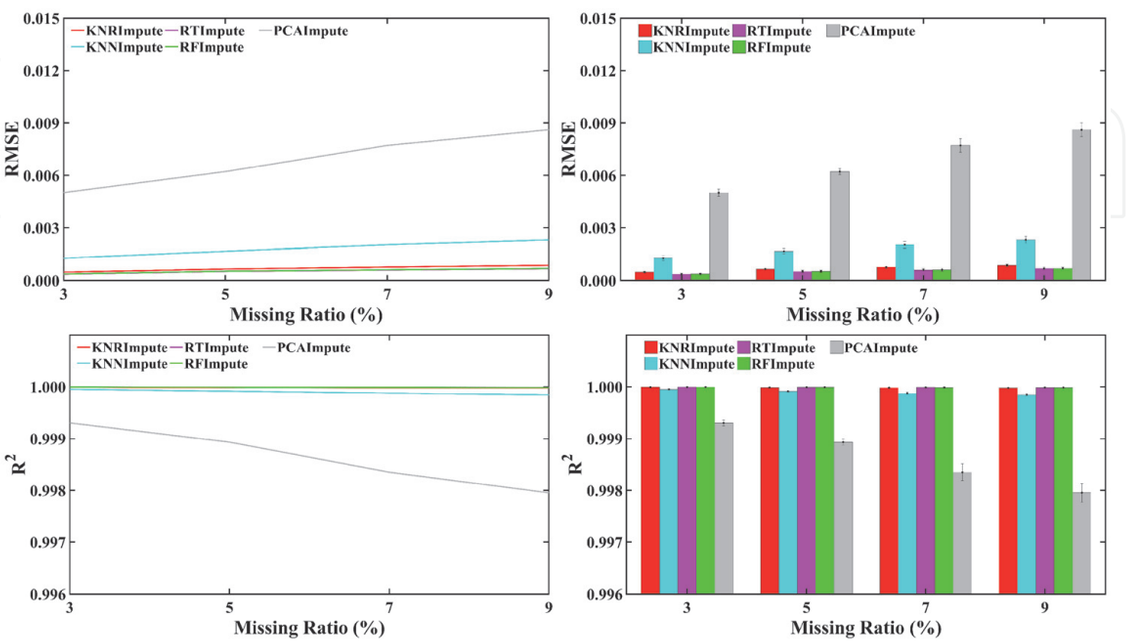


**Figure 4.**
*Average and the standard deviation of the RMSEs and $R^2$ based on dataset IP. The top ones are RMSEs, and the bottom are $R^2$.*

Besides, RMSEs became higher when missing rates were increased. Observations showed that KNRImpute, RTImpute, and RFImpute generated similar RMSEs. Overall, KNNImpute and PCAImpute were affected by the hyperparameters.

## 4. Conclusions

This chapter introduces recent methods for processing missing values. Besides, four types of commonly used algorithms, namely, *K*-Nearest Neighbors, regression, tree-based algorithms, and latent component-based approaches, were examined. Their advantages and disadvantages were also discussed in each subsection. It is worth noting that data imputation usually does not require training data. It becomes impractical when data imputation needs supervisory information or the ground truth (notably, the ground truth is unobservable). This is because when missing values occur in training data and even when the ground truth is missing, the supervised methods even cannot work to learn the ground truth. Therefore, those selected four types of commonly used algorithms in this chapter did not rely on and require any supervisory information.

To evaluate those commonly used algorithms, this chapter conducted experiments on open datasets. Criteria including root-mean-squared errors and coefficients of determination were adopted. Numerical results were also displayed in the experimental section for reference.

In more recent years, surveys showed that a deep learning model "Generative Adversarial Network (GAN)" has attracted much attention, and several novel imputation methods based on GANs have been proposed, e.g., MisGAN [49], MIWAE [50], and GAIN [51]. For future studies, deep learning architectures such as Deep PCA, PCANet, and Deep NMF, can be integrated into those four types of commonly used algorithms, namely, *K*-Nearest Neighbors, regression, tree-based algorithms, and latent component-based approaches and subsequently enhance data imputation.

## Acknowledgements

The authors would like to appreciate Guan-Yu Huang for conducting experiments to collect results. This work is supported in part by Pervasive Artificial Intelligence Research (PAIR) Labs, Taiwan.

*DOI: http://dx.doi.org/10.5772/intechopen.94068*



## Author details

Bo-Wei Chen[1] and Jia-Ching Wang[2*]

1 Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung City, Taiwan

2 Department of Computer Science and Information Engineering, National Central University, Taoyuan City, Taiwan

*Address all correspondence to: jcw@csie.ncu.edu.tw

## References

[1] J. Reilly, S. Dashti, M. Ervasti, J. D. Bray, S. D. Glaser, and A. M. Bayen, "Mobile phones as seismologic sensors: Automating data extraction for the iShake system," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 2, pp. 242–251, Apr. 2013.

[2] Z. Peng, S. Gao, B. Xiao, S. Guo, and Y. Yang, "CrowdGIS: Updating digital maps via mobile crowdsensing," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 369–380, Jan. 2018.

[3] L. Xu and Q. Huang, "EM estimation of nanostructure interactions with incomplete feature measurement and its tailored space filling designs," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 579–587, Jul. 2013.

[4] D. Li, Y. Zhou, G. Hu, and C. J. Spanos, "Handling incomplete sensor measurements in fault detection and diagnosis for building HVAC systems," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 833–846, Apr. 2020.

[5] S.-K. S. Fan, C.-Y. Hsu, D.-M. Tsai, F. He, and C.-C. Cheng, "Data-driven approach for fault detection and diagnostic in semiconductor manufacturing," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1925–1936, Oct. 2020

[6] M. M. Babu, "An introduction to microarray data analysis," in *Computational Genomics: Theory and Application*, R. P. Grant, Ed. UK: Taylor & Francis, 2004, pp. 225–249.

[7] T. Aittokallio, "Dealing with missing values in large-scale studies: Microarray data imputation and beyond," *Briefings in Bioinformatics*, vol. 11, no. 2, pp. 253–264, Mar. 2009.

[8] M. S. B. Sehgal, I. Gondal, and L. S. Dooley, "Collateral missing value imputation: A new robust missing value estimation algorithm for microarray data," *Bioinformatics*, vol. 21, no. 10, pp. 2417–2423, Feb. 2005.

[9] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[10] T. H. Bø, B. Dysvik, and I. Jonassen, "LSimpute: Accurate estimation of missing values in microarray data with least squares methods," *Nucleic Acids Research*, vol. 32, no. 3, Feb. 2004.

[11] C.-C. Chiu, S.-Y. Chan, C.-C. Wang, and W.-S. Wu, "Missing value imputation for microarray data: A comprehensive comparison study and a web tool," *BMC Systems Biology*, vol. 7, no. 6, Dec. 2013.

[12] E. D. de Leeuw, "Reducing missing data in surveys: An overview of methods," *Quality & Quantity*, vol. 35, no. 2, pp. 147–160, May 2001.

[13] S. Dorofeev, *Statistics for Real-life Sample Surveys: Non-simple-random Samples and Weighted Data*. Cambridge, UK: Cambridge University Press, Jul. 2006.

[14] K. P. Soman, S. Diwakar, and V. Ajay, *Insight into Data Mining: Theory and Practice*. India: Prentice-Hall, Apr. 2010.

[15] D. Bertsimas, C. Pawlowski, and Y. D. Zhuo, "From predictive methods to missing data imputation: An optimization approach," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 7133–7171, Jan. 2017.

[16] Y. C. Yuan, "Multiple imputation for missing data: Concepts and new development," SAS Institute Incorporation, Rockville, MD, Technical Report, 2000.

[17] T. H. Lin, "A comparison of multiple imputation with EM algorithm and MCMC method for quality of life missing data," *Quality & Quantity*, vol. 44, no. 2, pp. 277–287, Feb. 2010.

[18] D. I. Gad and B. R. Manjunatha, "Performance evaluation of predictive models for missing data imputation in weather data," in *Proc. 2017 International Conference on Advances in Computing, Communications and Informatics*, Udupi, India, 2017, Sep. 13–16, pp. 1327–1334.

[19] H. Kim, G. H. Golub, and H. Park, "Missing value estimation for DNA microarray gene expression data: Local least squares imputation," *Bioinformatics*, vol. 21, no. 2, pp. 187–198, Aug. 2004.

[20] Z. Cai, M. Heydari, and G. Lin, "Iterated local least squares microarray missing value imputation," *Journal of Bioinformatics and Computational Biology*, vol. 4, no. 5, pp. 935–957, Oct. 2006.

[21] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, "Random survival forests," *Annals of Applied Statistics*, vol. 2, no. 3, pp. 841–860, Sep. 2008.

[22] F. Tang and H. Ishwaran, "Random forest missing data algorithms," *Statistical Analysis and Data Mining*, vol. 10, no. 6, pp. 363–377, 2017.

[23] T. Marwala, *Computational Intelligence for Missing Data Imputation, Estimation, and Management: Knowledge Optimization Techniques*. Pennsylvania, USA: Information Science Reference, Mar. 2009.

[24] C. M. Salgado, C. Azevedo, H. Proença, and S. M. Vieira, "Missing data," in *Secondary Analysis of Electronic Health Records*, M. C. Data, Ed. Cham, Switzerland: Springer, 2016.

[25] L. P. Brás and J. C. Menezes, "Improving cluster-based missing value estimation of DNA microarray data," *Biomolecular Engineering*, vol. 24, no. 2, pp. 273–282, Jun. 2007.

[26] S. Zhang, "Nearest neighbor selection for iteratively kNN imputation," *Journal of Systems and Software*, vol. 85, no. 11, pp. 2541–2552, Nov. 2012.

[27] X. Zhang, X. Song, H. Wang, and H. Zhang, "Sequential local least squares imputation estimating missing value of microarray data," *Computers in Biology and Medicine*, vol. 38, no. 10, pp. 1112–1120, Oct. 2008.

[28] Z. Yu, T. Li, S.-J. Horng, Y. Pan, H. Wang, and Y. Jing, "An iterative locally auto-weighted least squares method for microarray missing value estimation," *IEEE Transactions on NanoBioscience*, vol. 16, no. 1, pp. 21–33, Jan. 2017.

[29] A. Wang, Y. Chen, N. An, J. Yang, L. Li, and L. Jiang, "Microarray missing value imputation: A regularized local learning method," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 3, pp. 980–993, May–Jun. 2019.

[30] M. S. Sehgal, I. Gondal, and L. Dooley, "Gene expression imputation techniques for robust post genomic knowledge discovery," in *Computational Intelligence in Medical Informatics*, A. Kelemen, A. Abraham, and Y. Liang, Eds. Berlin, Heidelberg, German: Springer-Verlag, 2009, pp. 185–206.

[31] X. Chen, Y. Cai, Q. Liu, and L. Chen, "Nonconvex $l_p$-norm regularized sparse self-representation for traffic sensor

data recovery," *IEEE Access*, vol. 6, pp. 24279–24290, 2018.

[32] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[33] D. J. Stekhoven and P. Bühlmann, "MissForest—Non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2011.

[34] K. A. Severson, M. C. Molaro, and R. D. Braatz, "Principal component analysis of process datasets with missing values," *Processes*, vol. 5, no. 3, pp. 38–55, Jul. 2017.

[35] B. Grung and R. Manne, "Missing values in principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 42, nos. 1–2, pp. 125–139, Aug. 1998.

[36] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proc*. *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, California, United States, 2007, Aug. 12–15, pp. 39–42.

[37] X. Kong, F. Xia, J. Wang, A. Rahim, and S. K. Das, "Time-location-relationship combined service recommendation based on taxi trajectory data," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1202–1212, Jun. 2017.

[38] C. Hsu, M. Yeh, and S. Lin, "A general framework for implicit and explicit social recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2228–2241, Dec. 2018.

[39] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the Netflix prize," in *Proc*. *4th International Conference on Algorithmic Applications in Management*, Shanghai, China, 2008, Jun. 23–25, pp. 337–348.

[40] L. Du, X. Li, and Y.-D. Shen, "Robust nonnegative matrix factorization via half-quadratic minimization," in *Proc. 12th IEEE International Conference on Data Mining*, Brussels, Belgium, 2012, Dec. 10–13, pp. 201–210.

[41] R. He, L. Wang, Z. Sun, Y. Zhang, and B. Li, "Information theoretic subspace clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 12, pp. 2643–2655, Dec. 2015.

[42] A. Liutkus, D. Fitzgerald, and R. Badeau, "Cauchy nonnegative matrix factorization," in *Proc*. *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, United States, 2015, Oct. 18–21.

[43] N. Guan, T. Liu, Y. Zhang, D. Tao, and L. S. Davis, "Truncated Cauchy non-negative matrix factorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 246–259, Jan. 2019.

[44] H. Wang, W. Yang, and N. Guan, "Cauchy sparse NMF with manifold regularization: A robust method for hyperspectral unmixing," *Knowledge-Based Systems*, vol. 184, Nov. 2019.

[45] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, Oct. 2007.

[46] Z. Yuan and E. Oja, "Projective nonnegative matrix factorization for image compression and feature extraction," in *Proc*. *Scandinavian Conference on Image Analysis*, Joensuu, Finland, 2005, Jun. 19–22, pp. 333–342.

[47] Z. Yang and E. Oja, "Linear and nonlinear projective nonnegative matrix

factorization," *IEEE Transactions on Neural Networks*, vol. 21, no. 5, pp. 734–749, May 2010.

[48] J. Wen, J. E. Fowler, M. He, Y.-Q. Zhao, C. Deng, and V. Menon, "Orthogonal nonnegative matrix factorization combining multiple features for spectral–spatial dimensionality reduction of hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 7, pp. 4272–4286, Jul. 2016.

[49] S. C.-X. Li, B. Jiang, and B. M. Marlin, "MisGAN: Learning from incomplete data with generative adversarial networks," in *Proc. 7th International Conference on Learning Representations*, New Orleans, Louisiana, USA, 2019, May 06–09.

[50] P.-A. Mattei and J. Frellsen, "MIWAE: Deep generative modelling and imputation of incomplete data sets," in *Proc. 36th International Conference on Machine Learning*, Long Beach, California, USA, 2019, Jun. 09–15, pp. 4413–4423.

[51] J. Yoon, J. Jordon, and M. v. d. Schaar, "GAIN: Missing data imputation using generative adversarial nets," in *Proc. 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018, Jul. 10–15, pp. 5675–5684.