Principle Components Analysis based frameworks for efficient missing data imputation algorithms

Anonymous Author(s)
Affiliation
Address
email

Abstract

Missing data is a commonly occurring problem in practice, and imputation, i.e., filling the missing entries of the data, is a popular way to deal with this problem. This motivates multiple works on imputation to deal with missing data of various types and dimensions. However, for high-dimensional datasets, these imputation methods can be computationally expensive. Therefore, in this work, we propose *Principle Component Analysis Imputation* (PCAI), a simple framework based on Principle Component Analysis (PCA) to speed up the imputation process of many available imputation techniques. Next, based on PCAI, we propose *PCA Imputation - Classification* (PIC), an imputation-dimension reduction-classification framework to deal with missing data classification problems where it is desirable to reduce the dimensions before training a classification model. Our experiments show that the proposed frameworks can be utilized with various imputation algorithms and improve the imputation speed significantly. Interestingly, the frameworks aid imputation methods that rely on many parameters by reducing the dimension of the data and hence, reducing the number of parameters needed to be estimated. Moreover, they not only can achieve compatible mean square error/higher classification accuracy compared to the traditional imputation style on the original missing dataset but many times deliver even better results. In addition, the frameworks also help to tackle the memory issue that many imputation approaches have by reducing the number of features.

1 Introduction

Despite recent efforts in directly handling missing data [1, 2, 3, 4], missing data imputation approaches [5, 6, 7] remain commonly used. This is because directly handling missing data can be complicated and usually are developed for specific target problems or models, while imputation can be more versatile. Specifically, an important advantage of imputation is that the imputed data becomes *complete*, i.e., no longer have any missing value. Therefore, it is easier to continue with other preprocessing steps, analysis, and data visualizations. Furthermore, one can deploy many models and choose the best one by using the available packages or software tools for non-missing data. Meanwhile, directly handling missing data strategies do not have these advantages. They are more complicated and not that readily available.

Many techniques have been developed for missing data imputation, ranging from traditional techniques such as MICE [5], K-Nearest Neighbors to recent machine learning/deep learning techniques such as GAIN [6], DL-GSA [7]. However, many of them can be computationally expensive for big datasets. For example, experiments in [8] show that under their experiment settings, for Fashion MNIST [9], a dataset of 70,000 samples and 784 features, the MICE [5] and missForest [10] techniques are unable to finish the imputation process within three hours of running for a missing rate

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

(the ratio between the number of missing entries versus the total number entries in the dataset) of 20%. Since datasets nowadays are trending towards larger sizes [11], with hundreds of thousands of features [12], it is necessary to speed up the available imputation techniques.

This work introduces two novel frameworks based on Principle Component Analysis (PCA) to speed up the imputation process of many available techniques or the imputation-classification process for missing data classification problems. The first framework, **PCA Imputation (PCAI)** is proposed to speed up the imputation speed by partitioning the data into the fully observed features partition and the partition of features with missing data. After that, the imputation of the missing part is performed based on the union of the PCA - reduced version of the fully observed part and the missing part. Interestingly, it turns out that the method has a great potential to aid the performance of methods that rely on many parameters, such as Deep Learning imputation techniques. Meanwhile, the second one, **PCA Imputation - Classification (PIC)** is proposed to deal with the missing data classification problems where dimension reduction is desirable in advance of the model training step. PIC is based on PCAI with some modifications. Note that these frameworks are different from the methods developed for principal component analysis under missing data presented in [13, 14], which are about how to conduct PCA when the data contains missing values.

In summary, the contributions of this article are as follows: (i) we introduce **PCAI** to improve the imputation speed of many available imputation techniques; (ii) we introduce **PIC** to deal with missing data classification problems where dimension reduction is desirable; (iii) we analyze the potential strength and drawbacks of these approaches; and (iv) we illustrate via experiments that our frameworks can work with various imputation strategies while achieve comparable or even lower mean square error/higher classification accuracies compared to the corresponding original approaches, and alleviate the memory issue in some approaches.

The rest of the paper is organized as follows. In Section 2, we review some related work in the field of missing data. Next, in Section 3 and Section 4, we introduce our novel PCAI and PIC frameworks, respectively. After that, in Section 5, we demonstrate their capabilities via experiments on various datasets. The paper ends with conclusions, remarks, and future works in Section 6.

2 Related Works

Various works have been done on missing data imputation to deal with different data analysis situations. As an example, if one is interested in modeling the uncertainty associated with the imputation, suitable approaches can be multiple or Bayesian imputation techniques such as multiple imputations using Deep Denoising Autoencoders [15], Bayesian Principal Component Analysis-based imputation [16], and extreme learning machine multiple imputation [17]. In addition, graphical models can be prominent candidates when transparency, estimability, and testability are desirable, and these approaches can provide meaningful performance guarantees even if the missing values are not at random [18]. Next, for continuous data, matrix completion techniques such as Fast Alternating Least Squares [19], softImpute [20] can quickly give good results. In biology, the missing values are often categorical, and the imputed values need to be interpretable. In such cases, classification techniques or tree-based methods such as decision trees and fuzzy clustering with iterative learning (DIFC) [21], missForest [10], the DMI algorithm [22], and sequential regression trees [23] are well-suited. In addition, some recently developed methods that can handle mixed missing data are SICE [24], FEMI [25], and HCMM-LD [26]. When the sample sizes are large enough compared to the number of features, deep learning techniques such as Multiple Imputation Using Deep Denoising Autoencoders [15], DL-GSA [7], and Swarm Intelligence-Deep Neural Network [27] can be powerful imputers. However, it is worth noting that deep learning methods usually require more data than statistical imputation approaches. Some other popularly used missing data imputation methods are multiple imputation by chained equation (MICE) [5], K-nearest Neighbors imputation (KNNI) [28], and mean imputation [28].

In addition, for the purpose of data imputation and data type, for classification, the impact of imputation techniques on different classifiers may vary. Specifically, [28] compares the performance of logistic regression with regularization, k-nearest neighbours (kNN), random forest, classification tree, and xgboost classifiers [28] on datasets with missing entries. They use different imputation methods (mean imputation/ MICE imputation [5]/ missForest [10]/ random imputation/ softImpute [20]/ hot deck imputation, kNN imputation) and compare the performance. According to the paper,

mean imputation seems to outperform other counterparts for logistic regression with regularization and kNN, random imputation wins for random forest, missForest seems to be the best imputer for classification tree, and hot deck imputation is the best for xgboost.

With the rapid growth of data size [11, 12], it is necessary to speed up the already developed imputation methods because many current approaches remain too slow for big datasets, as pointed out in an example in Section 1. This is where a popular dimension reduction method like PCA can come to use. PCA projects the original higher-dimensional dataset into a representation of lower dimensionality by extracting and retaining important information from the data and expressing this new information based on a set of orthogonal vectors known as principal components. Its goal is to find linear transformations of the original data that retain the maximal amount of variance. Note that there are some works on PCA under data missingness. For example, [13] considers the problem of finding principal components as an optimization problem of an objective function and proposes iterative solutions to it. On the other hand, [29] proposes a multiple imputation method for the estimates of the parameters (components and axes) of PCA to take into account the variability due to missing values. However, our work is different from these works in the sense that they target the problem of how to perform PCA for a dataset with missing data. Meanwhile, our frameworks utilize PCA to speed up the imputation processor to reduce the ratio between the number of features and the sample size.

3 PCA Imputation (PCAI)

In this section, we detail our PCAI framework, a PCA based framework that is capable of significantly improving the imputation speed of an imputer for high dimensional data, alleviating the memory issue for many approaches, and reducing the ratio between the number of features and number of samples to aid approaches with many parameters.

To start with some notations, let $pca(A)$ be a function of a data matrix A. The function returns $(\mathcal{R}_A, V)$ where $\mathcal{R}_A$ is the PCA-reduced version of A, and V is the projection matrix where the ith column of V is the eigenvector corresponding to the ith largest eigenvalue. Then, the framework is as depicted in Algorithm 1. We first conduct dimension reduction on the fully observed partition $\mathcal{M}$, which produces a reduced version $\mathcal{R}$ of $\mathcal{M}$. Then, the imputation of $\mathcal{M}$ is done on the set $\mathcal{R} \cup \mathcal{M}$ instead of $\mathcal{D} = \mathcal{F} \cup \mathcal{M}$ as how imputations are usually done (i.e., impute directly on the original missing data). In conducting dimension reduction, we expect to reduce the dimension of the fully observed partition so that the imputation of $\mathcal{M}$ can be faster.

Algorithm 1 PCAI framework

Require:
- $\mathcal{D} = \mathcal{F} \cup \mathcal{M}$ where $\mathcal{F}$ is the fully observed partition and $\mathcal{M}$ is the partition with missing values
- Imputer I

Procedure:
$(\mathcal{R}, V) \leftarrow pca(\mathcal{F})$
$\mathcal{M}' \leftarrow$ the imputed version of $\mathcal{M}$ based on $\mathcal{R} \cup \mathcal{M}$
return Imputed version $\mathcal{M}'$ of $\mathcal{M}$

One may reckon that using $\mathcal{R} \cup \mathcal{M}$ instead of $\mathcal{F} \cup \mathcal{M}$ may lead to loss of information due to dimension reduction and therefore lower the quality of imputation. However, as will be illustrated in the experiment sections, the differences between the mean squared error of the imputed version versus the ground truth for these approaches are only slightly different, and many times, PCAI seems to be slightly better. This is possibly because PCA retains the important information from the data while removing some noise. Thus, when that is the case, the imputation quality is improved in terms of MSE.

However, PCAI also has some shortcomings. For problems where the sample size n is smaller than the number of features in the fully observed block q, the covariance matrix has the size of $q \times q$, which is bigger than the size $n \times q$ of the fully observed partition. This may make the PCA dimension reduction process become computationally expensive, rendering PCAI to be slower than imputing directly on the original missing data. This issue will be illustrated in the experiment section.

4 PCA Imputation - Classification (PIC)

In many classification problems, it is desirable to conduct a dimension reduction before training a model. This is particularly of interest for problems where the number of features is much larger than the number of samples ($p \gg n$ problems). Since PCAI conducts PCA on the fully observed partition $\mathcal{F}$, it reduces the dimensions for a portion of the data. Therefore, rather than imputing values using the PCAI framework and then conducting a dimension reduction step on $\mathcal{F} \cup \mathcal{M}'$, one can perform dimension reduction on $\mathcal{M}'$ to get $\mathcal{R}'$, a PCA-reduced version of $\mathcal{M}'$. Then, one can use $\mathcal{F} \cup \mathcal{R}'$ as reduced dimension data. As will be shown in the experiments, this speeds up the imputation and classification process significantly. This is the basic idea of our PIC framework.

PIC operates as shown in Algorithm 2. The procedure starts by performing PCA on the training fully observed partition $\mathcal{F}_{train}$, which gives the reduced version $\mathcal{R}_{train}$ of $\mathcal{F}_{train}$ and a projection matrix V. Next, we project $\mathcal{F}_{test}$ on V to get the reduced version $\mathcal{R}_{test}$ of $\mathcal{F}_{test}$. Then, we impute $\mathcal{M}_{train}$ on $\mathcal{R}_{train} \cup \mathcal{M}_{train}$ to get the imputed version $\mathcal{M}'_{train}$. Next, we impute $\mathcal{M}_{test}$ on $\mathcal{R}_{test} \cup \mathcal{M}_{test}$ to get the imputed version $\mathcal{M}'_{test}$. After that, if $reduce_{miss}$ is set to true, we perform dimension reduction on $\mathcal{M}'_{train}, \mathcal{M}'_{test}$. Then, we train the classifier on $\mathcal{R}_{train} \cup \mathcal{R}'_{train}$, i.e., the union of the reduced version of $\mathcal{F}_{train}$ and the reduced version of $\mathcal{M}_{train}$. For prediction of a vector $\mathbf{x} \in \mathcal{D}$, we can decompose $\mathbf{x}$ into $\mathbf{x} = (\mathbf{x}_{\mathcal{F}}, \mathbf{x}_{\mathcal{M}})$. After that, we can project $\mathbf{x}_{\mathcal{F}}$ on V to get a projection $\mathbf{r}$. Similarly, we can project $\mathbf{x}_{\mathcal{M}}$ on W to a get projection $\mathbf{r}'$. Finally, we can predict the label of $\mathbf{x}$ using the classifier C with input $(\mathbf{r}, \mathbf{r}')$.

Note that $reduce_{miss}$ is an option. When the number of features in the missing partition is large, one may be interested in reducing the dimension of $\mathcal{M}'$, and therefore, set $reduce_{miss}$ to *True*. However, when the number of features in the missing partition is small, one may want to keep it to *False*.

Algorithm 2 PIC framework

Require:
- $\mathcal{D} = \mathcal{F} \cup \mathcal{M}$ where $\mathcal{F}$ is the fully observed partition and $\mathcal{M}$ is the partition with missing values
- $reduce_{miss} = True/False$: if *True*, perform dimension reduction on the imputed partitions; if *False*, do not perform dimension reduction on the imputed partitions
- $\mathcal{F}_{train}, \mathcal{F}_{test}$: the training and testing data of the fully observed partition $\mathcal{F}$, respectively
- $\mathcal{M}_{train}, \mathcal{M}_{test}$: the training and testing data of the partition that has missing data $\mathcal{M}$, respectively

- Imputer I, classifier C.

Procedure:
$(\mathcal{R}_{train}, V) \leftarrow pca(\mathcal{F}_{train})$
$R_{test} \leftarrow \mathcal{F}_{test} V$
$\mathcal{M}'_{train} \leftarrow$ imputed version of $\mathcal{M}_{train}$ based on $\mathcal{R}_{train} \cup \mathcal{M}_{train}$
$\mathcal{M}'_{test} \leftarrow$ imputed version of $\mathcal{M}_{test}$ based on $\mathcal{R}_{test} \cup \mathcal{M}_{test}$
if reduce$_{miss}$ **then**
  $(\mathcal{R}'_{train}, W) \leftarrow pca(\mathcal{M}'_{train})$
  $\mathcal{R}'_{test} \leftarrow \mathcal{M}'_{test} W$
  Train the classifier C based on $\mathcal{R}_{train} \cup \mathcal{R}'_{train}$
  Classify based on $\mathcal{R}_{test} \cup \mathcal{R}'_{test}$,
else
  Train the classifier based on $\mathcal{R}_{train} \cup \mathcal{M}'_{train}$
  Classify based on $\mathcal{R}_{test} \cup \mathcal{M}'_{test}$
end if
return trained classifier C

5 Experiments

5.1 Experiment settings

To illustrate the performance of our frameworks, we compare the results of PCAI with the traditional approach of imputing on the originally missing dataset, which we shall refer to as the **traditional** one. We compare their performance in terms of speed (seconds) and mean square error (MSE). The imputation approaches used for comparison are softImpute [20], MissForest [10] and Multiple

Table 1: Description of datasets used in our experiments

Dataset	# Classes	# Features	# Samples
Parkinson	2	754	756
Fashion MNIST	10	784	70000

Imputation by Chained Equation (MICE) [5], kNN, IterativeSVD [30]. We implement these methods via the *fancyimpute* [31], *scikit-learn* [32], and *missingpy*[1] packages, with default configurations. The codes will be available upon the acceptance of the paper. For PIC, we compare the 5-fold cross-validation score (the metric is accuracy) of PIC when dimension reduction is applied on the imputed missing part (**PIC reduce**), when dimension reduction is not applied on the imputed missing part (**PIC**), and when PCA is applied to the imputed version on the full missing data (**PCA on** $\mathcal{F} \cup \mathcal{M}'$). Here, the classifier is Support Vector Machine (SVM), and the number of principal components is chosen so that the minimum amount of variance explained by these components is 95%.

Table 2: (MSE, speed) for different PCAI approaches and traditional approaches on the Parkinson dataset.

		missing rate		
Imputer	Strategy	20%	40%	60%
softImpute	PCAI	(0.073, 0.860)	(0.185, 0.774)	(0.305, 0.875)
	traditional	(0.072, 4.097)	(0.188, 4.043)	(0.308, 4.467)
MICE	PCAI	(0.091, 139.811)	(0.186, 85.241)	(0.369, 109.815)
	traditional	NA	NA	NA
GAIN	PCAI	(0.254, 45.046)	(0.538, 43.938)	(0.779, 43.956)
	traditional	(0.608, 69.839)	(1.097, 70.548)	(1.369, 70.293)
missForest	PCAI	(0.064, 188.324)	(0.163, 178.849)	(0.292, 138.085)
	traditional	(0.058, 905.002)	(0.160, 692.150)	(0.258, 449.415)
KNNI	PCAI	(0.127, 0.355)	(0.299, 0.398)	(0.466, 0.416)
	traditional	(0.113, 0.310)	(0.274, 0.337)	(0.426, 0.372)

Table 3: (MSE, speed) for different PCAI approaches and traditional approaches on the Fashion MNIST dataset.

		missing rate		
Imputer	Strategy	20%	40%	60%
softImpute	PCAI	(0.032, 22.408)	(0.066, 22.797)	(0.109, 25.603)
	traditional	(0.032, 67.627)	(0.064, 69.349)	(0.107, 77.233)
MICE	PCAI	(0.027, 2218.864)	(0.055, 1374.558)	(0.095, 1641.962)
	traditional	NA	NA	NA
GAIN	PCAI	(0.053, 65.730)	(0.091, 68.752)	(0.137, 69.743)
	traditional	(0.041, 97.898)	(0.079, 99.049)	(0.125, 96.317)
missForest	PCAI	NA	NA	NA
	traditional	NA	NA	NA
KNNI	PCAI	(0.055, 1607.850)	(0.115, 2033.153)	(0.180, 2272.370)
	traditional	(0.049, 3042.752)	(0.102, 3659.300)	(0.161, 3959.832)

For illustration, the experiments are conducted on the Parkinson dataset [33] and the Fashion MNIST dataset [9]. Their details are available in Table 1. Additional experiments will be reported in the Appendix. Note that any datasets can be rearranged so that the first q features are not missing and the

[1] `https://pypi.org/project/missingpy/`

remaining ones are missing. Therefore, without loss of generality, we assume that the first q features of each dataset are not missing, and the remaining ones contain missing value(s). Then, we simulated missing data randomly on the missing partition with missing rates 20%, 40%, and 60%. Here, a missing rate of 20% means that 20% of the missing partition is missing. For both datasets, $q = 700$.

All experiments are run on an AMD Ryzen 7 3700X CPU with 8 Cores, 16 processing threads, 3.6GHz, and 16GB RAM except for KNNI in Table 5. The experiments for KNNI in Table 5 are run in Google Colaboratory[2]. We terminate an experiment if no result is produced after 6500 seconds of running or if there arises a memory allocating issue, and we denote this as **NA** in the result tables.

5.2 Results and Analysis

Table 4: (accuracy, speed) of SVM for different imputation-classification strategies on the Parkinson dataset.

		missing rate		
Imputer	Strategy	20%	40%	60%
softImpute	PIC reduce	(0.862, 1.026)	(0.862, 1.137)	(0.862, 1.161)
	PIC	(0.858, 1.008)	(0.858, 1.079)	(0.859, 1.112)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	(0.861, 4.116)	(0.862, 4.424)	(0.861, 4.718)
MICE	PIC reduce	(0.859, 204.605)	(0.861, 256.340)	(0.861, 240.211)
	PIC	(0.858, 524.739)	(0.859, 694.667)	(0.859, 925.426)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	NA	NA	NA
GAIN	PIC reduce	(0.857, 91.086)	(0.852, 102.861)	(0.848, 122.349)
	PIC	(0.851, 89.984)	(0.853, 104.773)	(0.853, 123.233)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	(0.855, 130.349)	(0.851, 149.864)	(0.851, 181.135)
missForest	PIC reduce	(0.859, 204.850)	(0.861, 276.537)	(0.858, 153.783)
	PIC	(0.858, 202.939)	(0.861, 277.067)	(0.858, 153.463)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	(0.861, 656.948)	(0.862, 729.872)	(0.861, 472.230)
KNNI	PIC reduce	(0.858, 0.533)	(0.861, 0.462)	(0.862, 0.625)
	PIC	(0.858, 0.513)	(0.861, 0.462)	(0.862, 0.607)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	(0.862, 0.696)	(0.862, 0.642)	(0.859, 0.803)

It is clear from the results in Table 2 that PCAI speeds up the imputation significantly in all imputation approaches used except KNNI. Specifically for the Parkinson dataset, at 20% missing rate, when PCAI is applied to missForest, the running time reduces to 188, 324 seconds, which is almost $1/5$ of the traditional approach (905, 002 seconds). Next, it is worth noticing from Table 3 that the traditional style of imputing on the original missing dataset gives NAs for all missing rates used, but applying the PCAI framework for MICE imputation does give a result.

Regarding the running time for KNNI, the running time for KNNI between the PCAI approach and the traditional methods for Parkinson (Table 2) is not much different. However, for the Fashion MNIST dataset, KNNI using the PCAI framework obviously deliver a competitive result in a significantly shorter time. Specifically, KNNI at a missing rate of 20% on MNIST gives a result after only 1607.850 seconds, while the traditional style of imputing on the missing dataset takes up to 3042.752 seconds. From these observations, we can draw out that KNNI with PCAI is slightly slower for Parkinson while much faster than the traditional imputing style on Fashion MNIST because Parkinson does not have as many samples as Fashion MNIST (Parkinson has 756 samples, while Fashion MNIST has 70000 samples).

Another interesting point to be drawn out from Table 2 is that PCAI approaches generate a lot of improvements in MSE for GAIN. For GAIN, this is possibly because PCA reduces the number of features while the sample size remains the same, making such a deep learning approach more applicable to the newly reduced data.

[2] `https://colab.research.google.com/`

Table 5: (accuracy, speed) of SVM for different imputation-classification strategies on the Fashion MNIST dataset.

		missing rate		
Imputer	Strategy	20%	40%	60%
softImpute	PIC reduce	(0.891, 285.707)	(0.890, 279.263)	(0.889, 353.506)
	PIC	(0.891, 593.652)	(0.891, 598.726)	(0.890, 609.252)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	(0.892, 467.406)	(0.891, 458.625)	(0.891, 486.162)
MICE	PIC reduce	(0.892, 2379.869)	(0.892, 1884.385)	(0.891, 2851.581)
	PIC	(0.892, 2416.485)	(0.891, 1907.529)	(0.892, 2897.839)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	NA	NA	NA
GAIN	PIC reduce	(0.892, 539.720)	(0.891, 580.754)	(0.891, 660.633)
	PIC	(0.891, 583.016)	(0.892, 608.384)	(0.891, 703.145)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	(0.891, 1320.018)	(0.891, 1351.854)	(0.890, 1573.656)
missForest	PIC reduce	NA	NA	NA
	PIC	NA	NA	NA
	PCA on $\mathcal{F} \cup \mathcal{M}'$	NA	NA	NA
KNNI	PIC reduce	(0.891, 3040.393)	(0.891, 4235.292)	(0.89, 6059.608)
	PIC	(0.891, 3110.041)	(0.891, 4279.305)	(0.889, 6088.542)
	PCA on $\mathcal{F} \cup \mathcal{M}'$	NA	NA	NA

Recall that *PIC reduce* refer to PIC when dimension reduction is applied on the imputed missing part, *PIC* refers to PIC when dimension reduction is not applied on the imputed missing part, and *PCA on* $\mathcal{F} \cup \mathcal{M}'$ refer to the case when PCA is applied to the imputed version on the full missing data. The result for experiments related to PIC are provided in Table 4 and Table 5. From the tables, it is clear that PIC approaches reduce the imputation time significantly while maintaining competitive classification accuracy compared to PCA on $\mathcal{F} \cup \mathcal{M}'$, in most of the cases. For example, at the missing rate 20% on the Parkinson dataset, when using PIC reduce with missForest imputer, the running time is 204.850s compared to the running time of 656.948s of PCA on $\mathcal{F} \cup \mathcal{M}'$. For KNNI and MICE on Fashion MNIST (Table 3), it is worth noticing that PCA on $\mathcal{F} \cup \mathcal{M}'$ cannot gives the result, but PIC approaches can. In addition, PCA on $\mathcal{F} \cup \mathcal{M}'$ with MICE imputer gives NAs on both the Parkinson and Fashion MNIST dataset, while the PIC approaches can give results within the limited time. This shows that PIC is a promising imputation framework that can speed up imputation algorithms and alleviate the memory issue.

6 Conclusion and Remarks

We have presented two novel frameworks, PCAI and PIC, that allow for more efficient handling of missing data. Our experiments showed that both frameworks may speed up imputation algorithms significantly while outperforming or having at least similar performance compared to existing methods in terms of MSE/accuracy. The frameworks can significantly improve the MSE/accuracy for imputation methods that rely on many parameters, such as deep learning methods or even least square methods in high dimensions. In addition, by reducing the dimensions, the frameworks can alleviate the memory issue for some imputation approaches such as MICE. Even though PIC is only introduced for classification, the same strategy can be applied to a regression problem. We would like to explore that in the future. In addition, since various dimension reduction techniques have been developed to suit different scenarios, it is worth investigating different dimension reduction techniques for PCAI and PIC.

References

[1] Zachary C Lipton, David C Kale, Randall Wetzel, et al. Modeling missing data in clinical time series with rnns. *Machine Learning for Healthcare*, 56, 2016.

[2] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

[3] Mostafa Mehdipour Ghazi, Mads Nielsen, Akshay Pai, M Jorge Cardoso, Marc Modat, Sebastien Ourselin, and Lauge Sørensen. Robust training of recurrent neural networks to handle missing data for disease progression modeling. *arXiv preprint arXiv:1808.05500*, 2018.

[4] Thu Nguyen, Duy H.M. Nguyen, Huy Nguyen, Binh T. Nguyen, and Bruce A. Wade. Epem: Efficient parameter estimation for multiple class monotone missing data. *Information Sciences*, 567:1–22, 2021.

[5] S van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, pages 1–68, 2010.

[6] Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pages 5689–5698. PMLR, 2018.

[7] Ayush Garg, Deepika Naryani, Garvit Aggarwal, and Swati Aggarwal. Dl-gsa: a deep learning metaheuristic approach to missing data imputation. In *International Conference on Sensing and Imaging*, pages 513–521. Springer, (2018).

[8] Thu Nguyen, Khoi Minh Nguyen-Duy, Duy Ho Minh Nguyen, Binh T. Nguyen, and Bruce Alan Wade. Dper: Direct parameter estimation for randomly missing data. *Knowledge-Based Systems*, 240:108082, 2022.

[9] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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

[11] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM computing surveys (CSUR)*, 50(6):1–45, 2017.

[12] Isabelle Guyon, Jiwen Li, Theodor Mader, Patrick A Pletscher, Georg Schneider, and Markus Uhr. Competitive baseline methods set new standards for the nips 2003 feature selection benchmark. *Pattern recognition letters*, 28(12):1438–1444, 2007.

[13] Bjørn Grung and Rolf Manne. Missing values in principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 42(1-2):125–139, 1998.

[14] Alexander Ilin and Tapani Raiko. Practical approaches to principal component analysis in the presence of missing values. *The Journal of Machine Learning Research*, 11:1957–2000, 2010.

[15] Lovedeep Gondara and Ke Wang. Multiple imputation using deep denoising autoencoders. *arXiv preprint arXiv:1705.02737*, 2017.

[16] Vincent Audigier, François Husson, and Julie Josse. Multiple imputation for continuous variables using a bayesian principal component analysis. *Journal of statistical computation and simulation*, 86(11):2140–2156, 2016.

[17] Dušan Sovilj, Emil Eirola, Yoan Miche, Kaj-Mikael Björk, Rui Nian, Anton Akusok, and Amaury Lendasse. Extreme learning machine for missing data using multiple imputations. *Neurocomputing*, 174:220–231, 2016.

[18] Karthika Mohan and Judea Pearl. Graphical models for processing missing data. *Journal of the American Statistical Association*, pages 1–42, 2021.

[19] Trevor Hastie, Rahul Mazumder, Jason D Lee, and Reza Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015.

[20] 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.

[21] Sanaz Nikfalazar, Chung-Hsing Yeh, Susan Bedingfield, and Hadi A Khorshidi. Missing data imputation using decision trees and fuzzy clustering with iterative learning. *Knowledge and Information Systems*, 62(6):2419–2437, 2020.

[22] Md Geaur Rahman and Md Zahidul Islam. Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques. *Knowledge-Based Systems*, 53:51–65, 2013.

[23] Lane F Burgette and Jerome P Reiter. Multiple imputation for missing data via sequential regression trees. *American journal of epidemiology*, 172(9):1070–1076, 2010.

[24] Shahidul Islam Khan and Abu Sayed Md Latiful Hoque. Sice: an improved missing data imputation technique. *Journal of big data*, 7(1):1–21, 2020.

[25] Md Geaur Rahman and Md Zahidul Islam. Missing value imputation using a fuzzy clustering-based em approach. *Knowledge and Information Systems*, 46(2):389–422, 2016.

[26] Jared S Murray and Jerome P Reiter. Multiple imputation of missing categorical and continuous values via bayesian mixture models with local dependence. *Journal of the American Statistical Association*, 111(516):1466–1479, 2016.

[27] Collins Leke and Tshilidzi Marwala. Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference on Swarm Intelligence*, pages 259–270. Springer, (2016).

[28] Katarzyna Woźnica and Przemysław Biecek. Does imputation matter? benchmark for predictive models. *arXiv preprint arXiv:2007.02837*, 2020.

[29] Julie Josse, François Husson, et al. Multiple imputation in principal component analysis. *Advances in data analysis and classification*, 5(3):231–246, 2011.

[30] 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.

[31] Alex Rubinsteyn and Sergey Feldman. fancyimpute: An imputation library for python. `https://github.com/iskandr/fancyimpute`, 2016.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[33] C Okan Sakar, Gorkem Serbes, Aysegul Gunduz, Hunkar C Tunc, Hatice Nizam, Betul Erdogdu Sakar, Melih Tutuncu, Tarkan Aydin, M Erdem Isenkul, and Hulya Apaydin. A comparative analysis of speech signal processing algorithms for parkinson's disease classification and the use of the tunable q-factor wavelet transform. *Applied Soft Computing*, 74:255–263, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] . See Table 2, Table 3, Table 4, Table 5 and Section 5.
   (b) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] .
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Did you describe the limitations of your work? [Yes] . See the last paragraph of Section 3.

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] . The code will be available in the form of a Github repository upon acceptance, to preserve anonymity.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . See Section 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . See Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] . See References list.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] . The code can be found on a github repository upon the acceptance of the paper.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]