

Blockwise Principal Component Analysis for monotone missing data imputation and dimensionality reduction

Tu T. Do, Mai Anh Vu, Tuan L. Vo
Dept. of Mathematics and Computer Science
University Of Science
Vietnam National University in Ho Chi Minh City
 Ho Chi Minh city, Vietnam

Hoang Thien Ly
Faculty of Mathematics and Information Science
Warsaw University of Technology
 Warsaw, Poland

Thu Nguyen, Steven Hicks, Pål Halvorsen, Michael A. Riegler
Department of Holistic Systems
Simula Metropolitan
 Oslo, Norway

Binh T. Nguyen
AISIA Research Lab
Department of Computer Science
University Of Science
Vietnam National University in Ho Chi Minh City
 Ho Chi Minh city, Vietnam

Abstract—Monotone missing data is a common problem in data analysis. However, imputation combined with dimensionality reduction can be computationally expensive, especially with the increasing size of datasets. To address this issue, we propose a Blockwise Principal Component Analysis Imputation (BPI) framework for dimensionality reduction and imputation of monotone missing data. The framework conducts Principal Component Analysis on the observed part of each monotone block of the data and then imputes on merging the obtained principal components using a chosen imputation technique. BPI can work with various imputation techniques and can significantly reduce imputation time compared to conducting dimensionality reduction after imputation. This makes it a practical and efficient approach for large datasets with monotone missing data. Our experiments validate the improvement in speed while achieving an accuracy that is comparable to the common strategy of imputation prior to dimensionality reduction.

Index Terms—missing data, monotone, dimensionality reduction.

I. INTRODUCTION

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of features in a dataset while preserving the essential information. It is an essential technique for several reasons, including simplifying the dataset's structure, reducing computational complexity, ameliorating overfitting issues, removing noise and redundancy in data, and enabling visualization because it helps transform the data into a lower-dimensional space to visualize and interpret. In addition, dimensionality reduction can improve the performance of machine learning models by reducing overfitting and increasing generalization ability.

However, if missing values exist in the dataset, it can be challenging to perform dimensionality reduction because many techniques require a complete dataset. In addition, monotone missing data (i.e., data where if a certain data point is missing,

then all subsequent data points in that sequence or series are also missing) is a common problem in practice [1], [2]. For example, in the following, $\mathcal{D}_1, \mathcal{D}_2$ are examples of datasets with monotone missing data patterns, while \mathcal{D}_3 is not.

$$\mathcal{D}_1 = \begin{pmatrix} 2 & 3 & 5 & 7 & 9 \\ 1 & 2 & 4 & * & * \\ 3 & 2 & 6 & * & * \end{pmatrix},$$

$$\mathcal{D}_2 = \begin{pmatrix} 8 & 3 & 5 & 7 & 1 \\ 1 & 2 & 4 & * & * \\ 3 & 2 & * & * & * \end{pmatrix},$$

$$\mathcal{D}_3 = \begin{pmatrix} 8 & 3 & 5 & 7 & 1 \\ 1 & 2 & 4 & * & * \\ 3 & 2 & * & 1 & 12 \end{pmatrix}.$$

The above is just a simple toy dataset for illustration. In practice, such a monotone pattern can happen in various scenarios. For example, in longitudinal studies [3], participants may miss follow-up assessments, which leads to the missingness of subsequent time points. Similarly, in various clinical trials where patients are monitored for a particular health outcome over a period of time, if a patient drops out of the study after certain visits, all subsequent data points for that patient are missing. This also creates a monotone missing pattern.

It is common to deal with missing data by filling in the missing entries via some imputation technique. Various imputation techniques [4]–[6] have been developed to deal with different scenarios and different types of data. However, many widely used imputation techniques are not suitable for large datasets. This is illustrated via experiments by Nguyen et al. [7], where MICE [6] and MissForest [5] are not even able

to finish imputation within three hours. Recently, PCAI [8] was introduced as a framework to speed up the imputation when there are many fully observed features in the data. Specifically, PCAI partitions 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 principal components of the fully observed and the missing part. However, even if imputation can be sped up, imputation and then conducting dimensionality reduction is still a computationally expensive approach. Hence, while it provides principle components directly from missing data, it is not a scalable approach.

In this work, we propose a block-wise principal component analysis Imputation (BPI) framework for dimensionality reduction for monotone missing data. BPI starts by conducting Principal Component Analysis (PCA) on the observed part of each block of the data and then imputes the data on the merge of the projection using some imputation technique. Since PCA is conducted on parts of the data before imputation, BPI significantly reduces the running time compared to the conventional strategy of imputing before reducing the dimension.

The main contributions are: (i) We introduce BPI, a novel framework for dimensionality reduction and imputation of missing data; (ii) We illustrate via experiments that BPI can work with various imputation methods and improve the running time significantly compared to conducting dimensionality reduction after imputation; (iii) We point out the drawbacks of our work and directions for future work.

The paper is structured as follows: Section II summarizes some related works in the field. Next, in Section III, we introduce our proposed approach, Blockwise PCA Imputation (BPI). Next, in Section IV, we present experimental results to demonstrate our approach's effectiveness and discuss the implications of these results. Finally, we conclude the paper and outline potential future research directions in Section V.

II. RELATED WORKS

Most works addressing the issue of missing data predominantly focus on imputation strategies that aim to substitute the absent entries with plausible values, ensuring that the data becomes complete before subsequent analyses. Initial traditional methodologies included mean, mode, and median imputation. These are simplistic approaches wherein the central tendency of the observed values is utilized to replace the missing ones. Another early technique was regression-based imputation, where missing values are predicted based on relationships with other variables in a regression-like manner.

With the advance of computers and machine learning, more advanced techniques were formulated. One such method is the k -nearest neighbors imputation (KNNI), which considers k similar instances (neighbors) from the dataset to compute a (weighted) average/majority voting as the imputed value. Additionally, decision tree-based techniques like missForest [5], DMI algorithm [9], and DIFC algorithm [10] were introduced. These strategies leverage the hierarchical structure of decision trees to handle missing data efficiently. Moreover,

matrix decomposition techniques such as Polynomial Matrix Completion [11] and SOFT-IMPUTE [12] also stand out. They use matrix factorization to exploit the inherent low-rank structure of the data and thereby infer the missing entries.

Next, Bayesian and multiple imputation methods offers promising results in missing data scenarios. Bayesian network imputation [13] leverages the probabilistic relationships between variables, while methods like multiple imputations using Deep Denoising Autoencoders [14] employ neural networks to learn complex patterns in the data and impute accordingly. Additionally, Bayesian principal component analysis-based imputation [15] combines the power of PCA with Bayesian inference to handle missing data.

Over the recent years, the rise of deep learning techniques has brought about a significant shift in how this challenge is approached [14], [16]–[20]. These techniques, leveraging intricate neural network architectures, have the capability to model complex patterns and relationships in data, making them particularly effective for imputation tasks. However, one of the trade-offs of employing deep learning is its insatiable appetite for data. In contrast to traditional statistical imputation techniques, deep learning models often require vast amounts of data to train effectively and avoid overfitting.

In recent years, many hybrid imputation techniques have also been developed. For example, HPM-MI [21] is a technique that uses K-means clustering to analyze various imputation techniques and apply the best one to a dataset. Another typical work is SvrFcmGa [22], where a fuzzy c-means clustering hybrid approach is combined with support vector regression and a genetic algorithm.

Moreover, some studies have also concentrated on imputation for monotone missing data. For example, [23] compares the performance of fully conditional specification imputation and multivariate normal imputation for monotone missing data with ordinal outcomes.

However, the scalability of imputation remains a problem that requires more work to deal with various data types and missing patterns [4]. Some works try to speed up the imputation process. For example, the PCAI framework [8] helps speed up an imputation algorithm by applying principal component analysis (PCA) on the features that have no missing entries. Next, the features with missing entries are imputed based on the union of itself with the principal components of the fully observed features. Their experiments show great performance in speed while maintaining competitive results compared to directly applying an imputation method. Later, the performance of the method for logistic regression was studied throughout in [24]. However, a limitation of PCAI is the margin of imputation speed gain is limited if the number of fully observed features is small.

Moreover, for large data sets, dimension techniques under missing data can be of great interest. In fact, there are some works on directly getting the principle components in PCA from missing data. For example, as discussed in [25], nonlinear iterative partial least squares algorithm (NIPALS), which performs iterative regressions with observed data [25],

or iterative imputation (IA) use iterative PCA models using predictions from previous models until convergence. As introduced in [25], several new methods for building (PCA) models that handle missing data by using IA adaption. Among those, IA combined with trimmed scores regression (TSR) combines PCA with multiple linear regression by trimming extreme values and regressing the remaining scores, showed promising performance. However, NIPALS may struggle with convergence if many values are missing, and IA may be time-consuming or require multiple iterations. However, the scalability of these methods has not been investigated throughout.

III. METHODOLOGY

When missing data is present in a dataset, it is a common practice to impute the data first before consequent processing and analysis. However, when a dataset is big, and dimension reduction is of interest, but the data contains missing values, it can be computationally expensive to impute the data as well. For monotone missing data, this burden can be ameliorated by applying PCA to the observed parts of each monotone block and imputing the union of the principle components obtained from each block. This is the basic idea of the BPI framework, which will now be detailed in this section. To start, assume that there is a dataset input \mathbf{x} of p features with the following monotone pattern:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_{11} & \dots & \mathbf{x}_{1n_k} & \dots & \mathbf{x}_{1n_3} & \dots & \mathbf{x}_{1n_2} & \dots & \mathbf{x}_{1n_1} \\ \mathbf{x}_{21} & \dots & \mathbf{x}_{2n_k} & \dots & \mathbf{x}_{2n_3} & \dots & \mathbf{x}_{2n_2} & \dots & * \\ \mathbf{x}_{31} & \dots & \mathbf{x}_{3n_k} & \dots & \mathbf{x}_{3n_3} & \dots & * & \dots & * \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{x}_{k1} & \dots & \mathbf{x}_{kn_k} & \dots & * & \dots & * & \dots & * \end{pmatrix}.$$

where $\mathbf{x}_{in_j} \in \mathbb{R}^{p_j \times 1}$ and each column represents an observation. That is, there are n_1 observations available on the first p_1 variables, n_2 observations available on the first $p_1 + p_2$ variables, and so on. Then the dimension of \mathbf{x} is $\sum_{j=1}^k p_j$ and one can partition the data into:

$$\begin{aligned} \mathbf{x}_1 &= (\mathbf{x}_{11} \dots \mathbf{x}_{1n_k} \dots \mathbf{x}_{1n_2} \dots \mathbf{x}_{1n_1}), \\ \mathbf{x}_2 &= (\mathbf{x}_{21} \dots \mathbf{x}_{2n_k} \dots \mathbf{x}_{2n_2}), \\ &\vdots \\ \mathbf{x}_k &= (\mathbf{x}_{k1} \dots \mathbf{x}_{kn_k}), \end{aligned}$$

where \mathbf{x}_1 has the size $p_1 \times n_1, \dots, \mathbf{x}_k$ has the size $p_k \times n_k$. Suppose that PCA upon \mathbf{x}_i gives

$$\begin{aligned} \mathbf{z}_1 &= (\mathbf{z}_{11} \dots \mathbf{z}_{1n_k} \dots \mathbf{z}_{1n_2} \dots \mathbf{z}_{1n_1}), \\ \mathbf{z}_2 &= (\mathbf{z}_{21} \dots \mathbf{z}_{2n_k} \dots \mathbf{z}_{2n_2}), \\ &\vdots \\ \mathbf{z}_k &= (\mathbf{z}_{k1} \dots \mathbf{z}_{kn_k}). \end{aligned}$$

Here, \mathbf{z}_i of size $q_i \times n_i, i = 1, \dots, k$.

Then, we can stack \mathbf{z}_i together, and insert empty entries (here, we denote each empty entry by $*$) to present missing values. This gives

$$\mathbf{z}^* = \begin{pmatrix} \mathbf{z}_{11} & \dots & \mathbf{z}_{1n_k} & \dots & \mathbf{z}_{1n_2} & \dots & \mathbf{z}_{1n_1} \\ \mathbf{z}_{21} & \dots & \mathbf{z}_{2n_k} & \dots & \mathbf{z}_{2n_2} & \dots & * \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{z}_{k1} & \dots & \mathbf{z}_{kn_k} & \dots & * & \dots & * \end{pmatrix} \quad (1)$$

Then, we can conduct imputation on \mathbf{z}^* to get an imputed version \mathbf{z} and consider it as an imputed reduced dimension version of \mathbf{x} . Since the imputation is conducted after stacking the observed reduced blocks. Therefore, there are fewer values to be imputed.

Note that not all the blocks are of large dimensions. For example, it is possible that while $\mathbf{x} \in \mathbb{R}^{1000}$ but $\mathbf{x}_2 \in \mathbb{R}^2$. In such a case, it may be preferable not to reduce the dimension of \mathbf{x}_2 .

Algorithm 1 BPI algorithm

Input:

- 1) partitions $\mathbf{x}_1, \dots, \mathbf{x}_k$ of the training input \mathbf{x} where $\mathbf{x}_i \in \mathbb{R}^{p_i \times n_i}$,
- 2) imputation algorithm I .

Procedure:

- 1: **for** $i = 1, \dots, k$: **do**
Apply PCA on \mathbf{x}_i gives $\mathbf{z}_i = (\mathbf{z}_{i1}, \dots, \mathbf{z}_{in_i}) \in \mathbb{R}^{q_i \times n_i}$
- 2: **end for**
- 3:

$$\mathbf{z}^* = \begin{pmatrix} \mathbf{z}_{11} & \dots & \mathbf{z}_{1n_k} & \dots & \mathbf{z}_{1n_2} & \dots & \mathbf{z}_{1n_1} \\ \mathbf{z}_{21} & \dots & \mathbf{z}_{2n_k} & \dots & \mathbf{z}_{2n_2} & \dots & * \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{z}_{k1} & \dots & \mathbf{z}_{kn_k} & \dots & * & \dots & * \end{pmatrix}$$

- 4: Impute \mathbf{z}^* by I gives \mathbf{z}

Return: \mathbf{z} as the imputed dimensional reduced version of \mathbf{x} .

With the above reasoning, the algorithm is formalized in Algorithm 1.

a) *Example:* As a toy example, suppose that we have a dataset where the input is

$$\mathbf{x} = \begin{pmatrix} 1 & 5 & 2 & 9 & 7 & 0 & 8 \\ 2 & 3 & 6 & 4 & 0 & 1 & 9 \\ 3 & 1 & 8 & 3 & 5 & 2 & 0 \\ 3 & 1 & 2 & 0 & 0 & * & * \\ 0 & 4 & 1 & 3 & 2 & * & * \\ 4 & 8 & 6 & * & * & * & * \\ 9 & 1 & 2 & * & * & * & * \end{pmatrix}.$$

Then one can partition the \mathbf{x} into:

$$\begin{aligned}\mathbf{x}_1 &= \begin{pmatrix} 1 & 5 & 2 & 9 & 7 & 0 & 8 \\ 2 & 3 & 6 & 4 & 0 & 1 & 9 \\ 3 & 1 & 8 & 3 & 5 & 2 & 0 \end{pmatrix}, \\ \mathbf{x}_2 &= \begin{pmatrix} 3 & 1 & 2 & 0 & 0 \\ 0 & 4 & 1 & 3 & 2 \end{pmatrix}, \\ \mathbf{x}_3 &= \begin{pmatrix} 4 & 8 & 6 \\ 9 & 1 & 2 \end{pmatrix}.\end{aligned}$$

Suppose that PCA upon \mathbf{x}_i gives \mathbf{z}_i as follows

$$\begin{aligned}\mathbf{z}_1 &= \begin{pmatrix} 0.5 & 2 & 1 & 0 & 1 & 0.9 & 2 \\ 1 & 0.7 & 0.3 & 2 & 0.5 & 1 & 1 \end{pmatrix}, \\ \mathbf{z}_2 &= \begin{pmatrix} 1 & 3 & 0.7 & 0 & 0.3 \end{pmatrix}, \\ \mathbf{z}_3 &= \begin{pmatrix} 2 & 0.5 & 1 \end{pmatrix}.\end{aligned}$$

Then, we can stack \mathbf{z}_i together and insert empty entries (here, we denote each empty entry by $*$) to present missing values. This gives

$$\mathbf{z}^* = \begin{pmatrix} 0.5 & 2 & 1 & 0 & 1 & 0.9 & 2 \\ 1 & 0.7 & 0.3 & 2 & 0.5 & 1 & 1 \\ 1 & 3 & 0.7 & 0 & 0.3 & * & * \\ 2 & 0.5 & 1 & * & * & * & * \end{pmatrix}. \quad (2)$$

We then can conduct imputation on \mathbf{z}^* , which has much fewer missing entries than \mathbf{x} .

A. Theoretical analysis

In this section, we analyze the explained variance of applying PCA on each block compared to PCA on data \mathbf{x} . Assume we have a data set \mathbf{x} with k blocks $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$ where \mathbf{x}_i has the size $p_i \times n$ ($1 \leq i \leq k$). Let \mathbf{S} denote the covariance matrix estimation of \mathbf{x} and \mathbf{S}_i denote the covariance matrix estimation of block i^{th} . Here, \mathbf{S}_i is a principal sub-matrix of \mathbf{S} . Now, considering i^{th} -block and suppose that applying PCA on \mathbf{x}_i will reduce the dimension from p_i to q_i ($< p_i$). So the explained variance of applying PCA upon \mathbf{x}_i is

$$\mathbf{EV}_{q_i}^{(i)} = \frac{\sum_{j=1}^{q_i} \lambda_j^{(i)}}{\sum_{j=1}^{p_i} \lambda_j^{(i)}} \cdot 100\% \quad (3)$$

where $\lambda_j^{(i)}$ is the j^{th} eigenvalue of non-increasing eigenvalues of \mathbf{S}_i , i.e. $\lambda_1^{(i)} \geq \lambda_2^{(i)} \geq \dots \geq \lambda_{p_i}^{(i)}$.

Let $q = q_1 + q_2 + \dots + q_k$. The explained variance of applying PCA on \mathbf{x} reduce the dimension from p to q presents

$$\mathbf{EV}_q = \frac{\sum_{j=1}^q \lambda_j}{\sum_{j=1}^p \lambda_j} \cdot 100\% \quad (4)$$

where λ_j is the j^{th} eigenvalue of non-increasing eigenvalues of \mathbf{S} . The lower bound and upper bound of the explained variance mean of all blocks are provided

$$k \cdot \frac{\lambda_{p-\min(p_i-q_i)}}{\sum_{j=1}^p \lambda_j} \leq \frac{1}{k} \sum_{i=1}^k \mathbf{EV}_{q_i}^{(i)} \leq 1 - k \cdot \frac{\lambda_p}{\sum_{j=1}^p \lambda_j} \quad (5)$$

Proof. For each block \mathbf{x}_i , note that \mathbf{S} is Hermitian and positive semidefinite matrix, and \mathbf{S}_i is a principal sub-matrix

of \mathbf{S} , by the well known Cauchy's interlacing theorem [26] we have

$$\lambda_j \geq \lambda_j^{(i)} \geq \lambda_{j+(p-p_i)}, \quad j = 1, 2, \dots, p_i. \quad (6)$$

$$\lambda_l \geq 0, \quad l = 1, 2, \dots, p. \quad (7)$$

Then,

$$\sum_{j=1}^{q_i} \lambda_j^{(i)} \geq \sum_{j=1}^{q_i} \lambda_{j+p-p_i} \geq \lambda_{q_i+p-p_i}, \quad i = 1, 2, \dots, k. \quad (8)$$

Therefore, for all $i \in \{1, 2, \dots, k\}$:

$$\sum_{j=1}^{q_i} \lambda_j^{(i)} \geq \lambda_{\max(q_i+p-p_i)} = \lambda_{p-\min(p_i-q_i)} \quad (9)$$

This implies that

$$\begin{aligned} \frac{1}{k} \sum_{i=1}^k \mathbf{EV}_{q_i}^{(i)} &= \frac{1}{k} \sum_{i=1}^k \frac{\sum_{j=1}^{q_i} \lambda_j^{(i)}}{\sum_{j=1}^{p_i} \lambda_j^{(i)}} \\ &\geq \frac{\lambda_{p-\min(p_i-q_i)}}{k} \sum_{i=1}^k \frac{1}{\sum_{j=1}^{p_i} \lambda_j^{(i)}} \end{aligned} \quad (10)$$

$$\geq \frac{\lambda_{p-\min(p_i-q_i)}}{k} \cdot \frac{k^2}{\sum_{i=1}^k \sum_{j=1}^{p_i} \lambda_j^{(i)}} \quad (11)$$

On the other hand, $\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_k$ are partition of \mathbf{S} so

$$\text{Tr}(\mathbf{S}_1) + \text{Tr}(\mathbf{S}_2) + \dots + \text{Tr}(\mathbf{S}_k) = \text{Tr}(\mathbf{S}) \quad (12)$$

where $\text{Tr}(\mathbf{S})$ is the sum of elements on the main diagonal of \mathbf{S} . By the property of eigenvalue,

$$\sum_{j=1}^{p_i} \lambda_j^{(i)} = \text{Tr}(\mathbf{S}_i), \quad i = 1, 2, \dots, k. \quad (13)$$

From (12) and (13) then

$$\sum_{i=1}^k \sum_{j=1}^{p_i} \lambda_j^{(i)} = \sum_{i=1}^k \text{Tr}(\mathbf{S}_i) = \text{Tr}(\mathbf{S}) = \sum_{j=1}^p \lambda_j \quad (14)$$

Thus, the inequality (11) becomes

$$\frac{1}{k} \sum_{i=1}^k \mathbf{EV}_{q_i}^{(i)} \geq k \cdot \frac{\lambda_{p-\min(p_i-q_i)}}{\sum_{j=1}^p \lambda_j} \quad (15)$$

For the upper bound,

$$\frac{1}{k} \sum_{i=1}^k \mathbf{EV}_{q_i}^{(i)} = \frac{1}{k} \sum_{i=1}^k \frac{\sum_{j=1}^{q_i} \lambda_j^{(i)}}{\sum_{j=1}^{p_i} \lambda_j^{(i)}} \quad (16)$$

$$= \frac{1}{k} \sum_{i=1}^k \left(1 - \frac{\sum_{j=q_i+1}^{p_i} \lambda_j^{(i)}}{\sum_{j=1}^{p_i} \lambda_j^{(i)}} \right) \quad (17)$$

$$= 1 - \frac{1}{k} \sum_{i=1}^k \frac{\sum_{j=q_i+1}^{p_i} \lambda_j^{(i)}}{\sum_{j=1}^{p_i} \lambda_j^{(i)}} \quad (18)$$

Furthermore, for all $i \in \{1, 2, \dots, k\}$ we get

$$\sum_{j=q_i+1}^{p_i} \lambda_j^{(i)} \geq \sum_{j=q_i+1}^{p_i} \lambda_{j+(p-p_i)} \geq \lambda_p \quad (19)$$

Together, (18) and (19) imply that

$$\frac{1}{k} \sum_{i=1}^k \mathbf{EV}_{q_i}^{(i)} \leq 1 - \frac{\lambda_p}{k} \sum_{i=1}^k \frac{1}{\sum_{j=1}^{p_i} \lambda_j^{(i)}} \quad (20)$$

$$\leq 1 - \frac{\lambda_p}{k} \cdot \frac{k^2}{\sum_{i=1}^k \sum_{j=1}^{p_i} \lambda_j^{(i)}} \quad (21)$$

By (14) then

$$\frac{1}{k} \sum_{i=1}^k \mathbf{EV}_{q_i}^{(i)} \leq 1 - k \cdot \frac{\lambda_p}{\sum_{j=1}^p \lambda_j} \quad (22)$$

The inequality (5) shows that the upper bound and lower bound depend on the number of blocks of data \mathbf{x} . When k is higher, the upper bound of the explained variance mean will be smaller. Therefore, we should consider applying the strategy of PCA upon each block of data if the upper bound is smaller than the expectation of explained variance.

IV. EXPERIMENTS

A. Experimental setting

To illustrate that BPI can work with various imputation techniques, we employed GAIN [27], and Soft Impute [12] as imputation methods. The experiments are conducted on three datasets: MNIST [28], Fashion MNIST [29], each of 10 classes, 784 features, and 70000 samples. In addition, we also use the RNA-Seq (HiSeq) PANCAN dataset [30], a dataset of 5 classes, where the number of features is significantly higher than the number of samples (20531 features versus 801 samples). For each dataset, we emulate the monotone missing pattern. For example, on the MNIST dataset, we remove the pixels at the bottom right corner of each image. We selected 6000 samples for each Fashion MNIST, MNIST, and the entire PANCAN dataset. We emulated the monotone missing condition for Fashion MNIST and MNIST datasets by dividing the dataset into four partitions and removing 100, 200, and 300 features for the second and third partitions, respectively. For the PANCAN dataset, the number of missing features for the second, third, and fourth partitions is 2000, 4000, and 6000, respectively. A summary of the datasets and the number of missing features can be found in table I.

All experiments were done on a computer with 16GB of RAM and 6 CPU cores, running the Linux operating system on an Intel I5-9400F, 4.100 Ghz. The link to the source codes will be provided upon the acceptance of the paper. For the results, we refer to the **baseline** when the data is imputed directly with an imputation method, and then PCA is applied to the imputed data, and finally, a classifier is trained on the resulting data. Meanwhile, **BPI** refers to imputing data based on the BPI framework along with an imputation technique and then training a classifier on the resulting dataset.

B. Result & Analysis

Across all datasets, we observed significant reductions in the imputation time of BPI compared to the corresponding baseline, ranging from 52% to 88%. In particular, reductions

for the Fashion MNIST datasets are 61% and 76% for GAIN and SOFT IMPUTE methods, respectively. For the MNIST dataset, the reductions are 52% and 87% for the two methods. Notably, for the RNA-Seq (HiSeq) PANCAN dataset, we observed an 88% reduction in imputation time for SOFT IMPUTE. A common trend among the three datasets is that we observed the most significant reduction in imputation time with Soft Impute and the least significant reduction in the GAIN method. Notably, while imputing the RNA-Seq (HiSeq) PANCAN dataset with baseline Soft Impute took 100 seconds; it only took 12 seconds for BPI Soft Impute.

Interestingly, with the same imputation method on the RNA-Seq (HiSeq) PANCAN dataset, we see an improvement of 0.3% and 2.8% for BPI compared to the corresponding baseline on the KNN and SVM classifier and a slight decrease of 0.3% on the Neural Network Classifier. Note that this is a dataset where the number of features is significantly higher than the number of observations. On the other hand, for the GAIN imputation on the Fashion MNIST dataset, we observed a decrease of 5.4%, 8.1%, and 4.5% of the BPI method compared to the corresponding baseline for the KNN, Neural Network, and SVM classifier. With the same imputation method on the MNIST dataset, we observed a 7% and 5% decrease for the KNN and Neural Network classifier and a 1.7% improvement in the SVM classifier. For the Soft Impute method on the Fashion MNIST dataset, we see a reduction of 6.3%, 1.86%, and 1.87% in accuracy for the KNN, Neural Network, and SVM classifier, respectively.

For the MNIST dataset, we see a drop of 6.2% and 0.1% for the KNN and Neural Network classifiers and an improvement of 2.6% on the SVM classifier. In summary, we see a slight compromise in accuracy trade-off for a significant reduction in imputation time for the BPI framework. More details of the results are reported in table II.

V. CONCLUSION

In this paper, we introduced BPI, a novel dimensionality reduction-imputation framework that can speed up the running time significantly compared to using PCA after imputing the missing data. This efficiency gain is particularly important in scenarios where time constraints are a crucial factor, especially for high-dimensional datasets. However, it is important to note that our proposed technique has certain limitations. Specifically, a drawback of the proposed technique is that PCA requires continuous data. For categorical data, a potential solution is to use one-hot encoding or learned embedding, and we will perform BPI for categorical data in the future. Additionally, since PCA has the denoising property, it may improve the imputation quality or classification accuracy for noisy datasets. In the future, we would like to investigate this through noisy datasets.

REFERENCES

- [1] T. Nguyen, D. H. Nguyen, H. Nguyen, B. T. Nguyen, and B. A. Wade, "Epem: Efficient parameter estimation for multiple class monotone missing data," *Information Sciences*, vol. 567, pp. 1–22, 2021.

TABLE I

DATASETS UNDER STUDY. THE NUMBER OF MISSING FEATURES IS A TUPLE OF 3, THE FIRST, SECOND, AND THIRD ELEMENT BEING THE NUMBER OF MISSING FEATURES OF THE DATASETS' SECOND, THIRD, AND FOURTH PARTITIONS, RESPECTIVELY.

Dataset	# Classes	# Features	# Samples	# Missing feature
MNIST [28]	10	784	6000	(100, 200, 300)
Fashion MNIST [29]	10	784	6000	(100, 200, 300)
PANCAN [30]	5	20531	801	(2000, 4000, 6000)

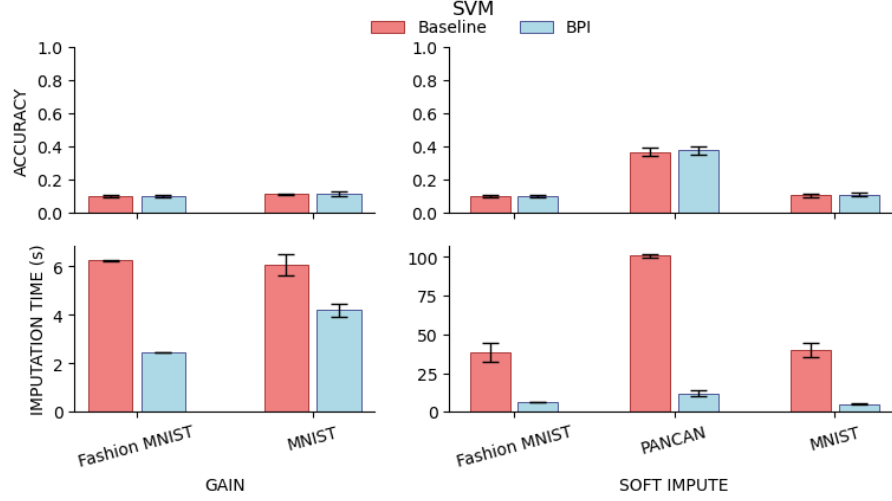


Fig. 1. Time and accuracy comparison between **Baseline** and BPI with different imputation methods across various datasets using SVM classifier.

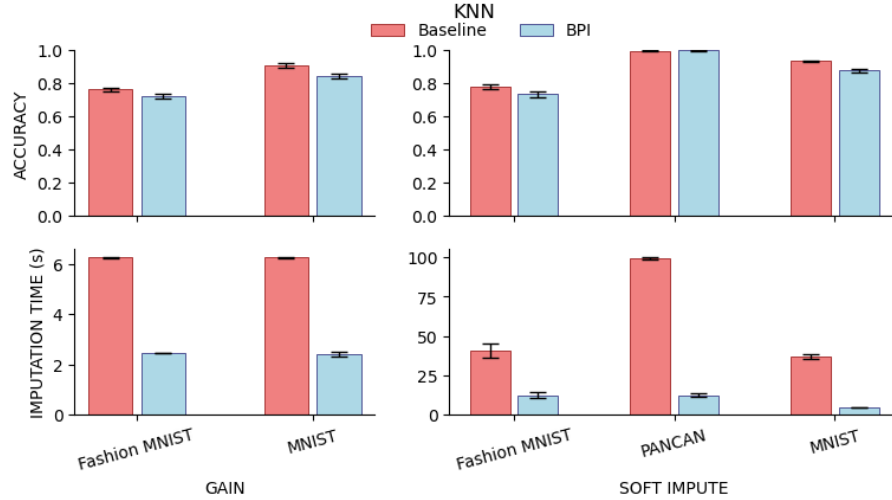


Fig. 2. Time and accuracy comparison between **Baseline** and BPI with different imputation methods across various datasets using KNN classifier.

- [2] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*. John Wiley & Sons, 2019, vol. 793.
- [3] D. Hedeker and R. D. Gibbons, *Longitudinal data analysis*. John Wiley & Sons, 2006.
- [4] M. A. Vu, T. Nguyen, T. T. Do, N. Phan, P. Halvorsen, M. A. Riegler, and B. T. Nguyen, "Conditional expectation for missing data imputation," *arXiv preprint arXiv:2302.00911*, 2023.
- [5] D. J. Stekhoven and P. Bühlmann, "Missforest-non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2012.
- [6] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, pp. 1–68, 2010.
- [7] T. Nguyen, K. M. Nguyen-Duy, D. H. M. Nguyen, B. T. Nguyen, and B. A. Wade, "Dper: Direct parameter estimation for randomly missing data," *Knowledge-Based Systems*, vol. 240, p. 108082, 2022.
- [8] T. Nguyen, H. T. Ly, M. A. Riegler, and P. Halvorsen, "Principle components analysis based frameworks for efficient missing data imputation algorithms," *arXiv preprint arXiv:2205.15150*, 2022.
- [9] M. G. Rahman and M. Z. Islam, "Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques," *Knowledge-Based Systems*, vol. 53, pp. 51–65, 2013.
- [10] S. Nikfalazar, C.-H. Yeh, S. Bedingfield, and H. A. Khorshidi, "Missing

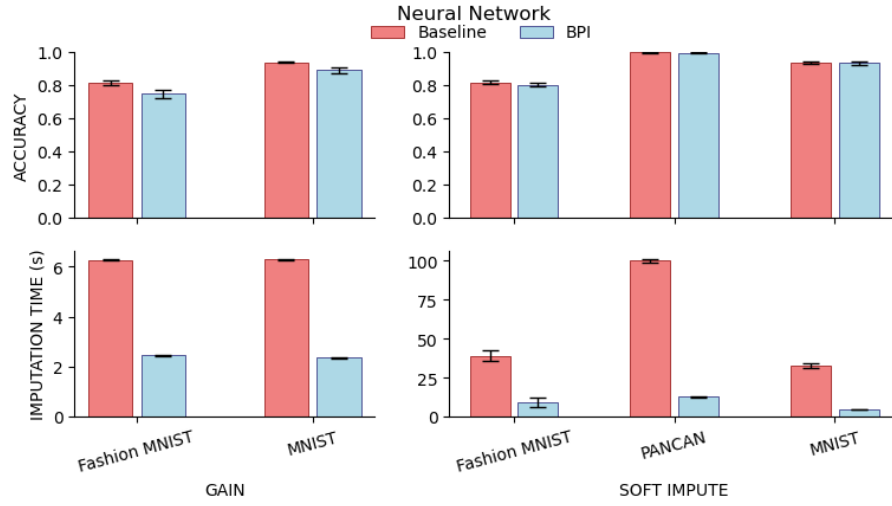


Fig. 3. Time and accuracy comparison between **Baseline** and **BPI** with different imputation methods across various datasets using a Neural Network classifier.

TABLE II

A SUMMARY OF CLASSIFICATION ACCURACY AND IMPUTATION TIME COMPARISON BETWEEN THE BASELINE AND BPI ACROSS DIFFERENT DATASETS, CLASSIFIERS, AND IMPUTATION METHODS. EACH EXPERIMENT WAS REPEATED 10 TIMES TO CALCULATE THE MEAN AND STANDARD DEVIATION FOR ACCURACY AND IMPUTATION TIME. FOR THE THE RNA-SEQ (HiSeq) PANCAN DATASET, SINCE THE NUMBER OF FEATURES IS HIGHER THAN THE NUMBER OF SAMPLES, WE DID NOT PERFORM GAIN IN THE BASELINE METHOD.

Imputer	Dataset	Classifier	Accuracy		Imputation time (second)	
			Baseline	BPI	Baseline	BPI
GAIN	Fashion MNIST	KNN	0.762 ± 0.009	0.721 ± 0.015	6.253 ± 0.016	2.444 ± 0.011
		Neural Network	0.814 ± 0.014	0.748 ± 0.025	6.261 ± 0.030	2.444 ± 0.020
		SVM	0.104 ± 0.007	0.100 ± 0.007	6.244 ± 0.034	2.450 ± 0.017
	PANCAN	KNN	NA	0.995 ± 0.005	NA	11.938 ± 0.496
		Neural Network	NA	0.946 ± 0.045	NA	11.850 ± 0.570
		SVM	NA	0.368 ± 0.023	NA	12.310 ± 0.651
	MNIST	KNN	0.908 ± 0.011	0.844 ± 0.013	6.245 ± 0.020	2.400 ± 0.099
		Neural Network	0.939 ± 0.005	0.891 ± 0.015	6.279 ± 0.022	2.363 ± 0.018
		SVM	0.114 ± 0.003	0.116 ± 0.012	6.083 ± 0.411	4.200 ± 0.244
Soft Impute	Fashion MNIST	KNN	0.783 ± 0.013	0.734 ± 0.017	40.939 ± 4.288	12.346 ± 1.859
		Neural Network	0.818 ± 0.009	0.803 ± 0.010	39.040 ± 3.387	9.278 ± 2.820
		SVM	0.102 ± 0.009	0.100 ± 0.006	38.328 ± 5.663	6.223 ± 0.103
	PANCAN	KNN	0.995 ± 0.004	0.998 ± 0.003	99.574 ± 0.629	12.293 ± 1.095
		Neural Network	0.999 ± 0.002	0.996 ± 0.004	100.010 ± 0.964	12.793 ± 0.411
		SVM	0.369 ± 0.026	0.380 ± 0.024	100.990 ± 0.967	11.894 ± 1.692
	MNIST	KNN	0.934 ± 0.004	0.877 ± 0.009	36.716 ± 1.352	4.759 ± 0.061
		Neural Network	0.936 ± 0.008	0.934 ± 0.009	32.907 ± 1.361	4.799 ± 0.097
		SVM	0.108 ± 0.010	0.110 ± 0.011	40.181 ± 4.401	5.001 ± 0.337

data imputation using decision trees and fuzzy clustering with iterative learning,” *Knowledge and Information Systems*, vol. 62, no. 6, pp. 2419–2437, 2020.

- [11] J. Fan, Y. Zhang, and M. Udell, “Polynomial matrix completion for missing data imputation and transductive learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, (2020), pp. 3842–3849.
- [12] R. Mazumder, T. Hastie, and R. Tibshirani, “Spectral regularization algorithms for learning large incomplete matrices,” *Journal of machine learning research*, vol. 11, no. Aug, pp. 2287–2322, 2010.
- [13] E. R. Hruschka, E. R. Hruschka, and N. F. Ebecken, “Bayesian networks for imputation in classification problems,” *Journal of Intelligent Information Systems*, vol. 29, no. 3, pp. 231–252, 2007.
- [14] L. Gondara and K. Wang, “Multiple imputation using deep denoising autoencoders,” *arXiv preprint arXiv:1705.02737*, 2017.
- [15] V. Audigier, F. Husson, and J. Josse, “Multiple imputation for continuous variables using a bayesian principal component analysis,” *Journal of statistical computation and simulation*, vol. 86, no. 11, pp. 2140–2156, 2016.
- [16] F. V. Nelwamondo, D. Golding, and T. Marwala, “A dynamic pro-

gramming approach to missing data estimation using neural networks,” *Information Sciences*, vol. 237, pp. 49–58, 2013.

- [17] S. J. Choudhury and N. R. Pal, “Imputation of missing data with neural networks for classification,” *Knowledge-Based Systems*, vol. 182, p. 104838, 2019.
- [18] A. Garg, D. Naryani, G. Aggarwal, and S. Aggarwal, “DI-gsa: a deep learning metaheuristic approach to missing data imputation,” in *International Conference on Sensing and Imaging*. Springer, 2018, pp. 513–521.
- [19] C. Leke and T. Marwala, “Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach,” in *International Conference on Swarm Intelligence*. Springer, 2016, pp. 259–270.
- [20] K. Mohan and J. Pearl, “Graphical models for processing missing data,” *Journal of the American Statistical Association*, pp. 1–42, 2021.
- [21] A. Purwar and S. K. Singh, “Hybrid prediction model with missing value imputation for medical data,” *Expert Systems with Applications*, vol. 42, no. 13, pp. 5621–5631, 2015.
- [22] I. B. Aydilek and A. Arslan, “A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm,” *Information Sciences*, vol. 233, pp. 25–35, 2013.
- [23] A. Y. Kombo, H. Mwambi, and G. Molenberghs, “Multiple imputation for ordinal longitudinal data with monotone missing data patterns,” *Journal of Applied Statistics*, vol. 44, no. 2, pp. 270–287, 2017.
- [24] T. H. Nguyen, B. Le, P. Nguyen, L. G. Tran, T. Nguyen, and B. T. Nguyen, “Principal components analysis based imputation for logistic regression,” in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2023, pp. 28–36.
- [25] A. Folch-Fortuny, F. Arteaga, and A. Ferrer, “Pca model building with missing data: New proposals and a comparative study,” *Chemometrics and Intelligent Laboratory Systems*, vol. 146, pp. 77–88, 2015.
- [26] M. S. Gowda and J. Tao, “The cauchy interlacing theorem in simple euclidean jordan algebras and some consequences,” *Linear and Multilinear Algebra*, vol. 59, no. 1, pp. 65–86, 2011.
- [27] J. Yoon, J. Jordon, and M. Schaar, “Gain: Missing data imputation using generative adversarial nets,” in *International conference on machine learning*. PMLR, 2018, pp. 5689–5698.
- [28] Y. LeCun, “The mnist database of handwritten digits,” <http://yann.lecun.com/exdb/mnist/>, 1998.
- [29] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [30] S. Fiorini, “gene expression cancer RNA-Seq,” UCI Machine Learning Repository, 2016, DOI: <https://doi.org/10.24432/C5R88H>.