Data Mining and Knowledge Discovery (2020) 34:1859–1897
https://doi.org/10.1007/s10618-020-00706-8

MIDIA: exploring denoising autoencoders for missing data imputation

Qian Ma[1] · **Wang-Chien Lee**[2] · **Tao-Yang Fu**[2] · **Yu Gu**[3] · **Ge Yu**[3]

Received: 16 August 2019 / Accepted: 17 July 2020 / Published online: 25 July 2020

Abstract

Due to the ubiquitous presence of missing values (MVs) in real-world datasets, the MV imputation problem, aiming to recover MVs, is an important and fundamental data preprocessing step for various data analytics and mining tasks to effectively achieve good performance. To impute MVs, a typical idea is to explore the correlations amongst the attributes of the data. However, those correlations are usually complex and thus difficult to identify. Accordingly, we develop a new deep learning model called *MIssing Data Imputation denoising Autoencoder* (MIDIA) that effectively imputes the MVs in a given dataset by exploring non-linear correlations between missing values and non-missing values. Additionally, by considering various data missing patterns, we propose two effective MV imputation approaches based on the proposed MIDIA model, namely MIDIA-Sequential and MIDIA-Batch. MIDIA-Sequential imputes the MVs attribute-by-attribute sequentially by training an independent MIDIA model for each incomplete attribute. By contrast, MIDIA-Batch imputes the MVs in one batch by training a uniform MIDIA model. Finally, we evaluate the proposed approaches by experimentation in comparison with existing MV imputation algorithms. The experimental results demonstrate that both MIDIA-Sequential and MIDIA-Batch achieve significantly higher imputation accuracy compared with existing solutions, and the proposed approaches are capable of handling various data missing patterns and data types. Specifically, MIDIA-Sequential performs better than MIDIA-Batch for data with monotone missing pattern, while MIDIA-Batch performs better than MIDIA-Sequential for data with general missing pattern.

Keywords Missing data imputation · Denoising autoencoder · MIDIA · Deep learning

Responsible editor: Shuiwang Ji.

✉ Qian Ma
maqian@dlmu.edu.cn

Extended author information available on the last page of the article

1 Introduction

Due to various uncontrollable factors, e.g., hardware failure, unconscious malfunction, participants refusal, etc, missing values (MVs) widely exist in various kinds of real-world datasets, e.g., medical datasets, microarray gene datsets, survey datasets and sensing datasets. To many algorithms employed in data analytics, data mining and machine learning (Gharibshah et al. 2020; Dong et al. 2014), data integrity is a prerequisite due to the incompetence of these algorithms in handling datasets with MVs. Moreover, the existence of MVs resulting in information loss, may cause performance degradation of the employed algorithms (Anagnostopoulos and Triantafillou 2014). Therefore, the critical task of *missing value imputation* (MV imputation), aiming to replace the MVs with some plausible estimations, attracts much research attention from the academia and industry.

Over years, various MV imputation methods have been proposed. Several existing works propose to estimate the missing value on an attribute of a data record by taking a weighted mean of values on the same attribute of some *similar* data records, e.g., hot-deck imputation (Andridge and Little 2010; Joenssen and Bankhofer 2012) and kNN imputation (Aittokallio 2010; Zhang 2008), which define some similarity functions and impute MVs by top-k similar data records. However, determining a proper similarity function and a suitable size of similar record set are very difficult. In a different line of research on MV imputation, some existing methods explore the *correlations* amongst attributes of the same data record. Among them, owing to the low computational cost, linear regression model (Wang and Rao 2002a) is often proposed to impute MVs by modeling linear correlations between *incomplete attributes* (attributes with MVs) and *complete attributes* (attributes without MVs). Nevertheless, the correlations amongst attributes in real-world datasets may be complex and hard to capture precisely using a linear model, e.g., a strong non-linear correlation has been found in some gene datasets (Zhou et al. 2003). To the best knowledge of the authors, few works on MV imputation effectively capture the non-linear correlations amongst attributes. Existing non-linear regression models are mostly based on kernel functions, e.g., Gaussian kernel, Uniform kernel and Logistic kernel, guided by experience (Zhu et al. 2011; Qin et al. 2009). They suffer the same problem with the aforementioned linear regression models since it is hard to select a proper kernel function to capture the complex interactions of various factors captured in the data.

Along with the advances in computer hardware, ever-increasing computing power, and many promising real-life applications, deep learning and neural networks (NN) have received tremendous attention in recent years. Among the various well-known NN models, *AutoEncoder* (AE) first encodes a data record into a low-dimensional latent vector which in turn is decoded back to the original data record in order to embed the inherent properties and the (usually non-linear) correlations amongst attributes into a latent vector. More specifically, by training the network to minimize a distortion measure between inputs and outputs (both are the input data record itself), an AE aims to learn a low-dimensional latent vector (called *embedding*) of the data record, to obtain a property-preserving representation of the raw input data record. Furthermore, to achieve a *good* embedding of the data for handling the issues of noisy data and data sparsity, *denoising AutoEncoder* (dAE) (Vincent et al. 2010, 2008), a stochastic version of autoencoder is proposed. Specifically, in order to obtain a robust embedding and avoid simply learning the identity, a dAE takes a partially corrupted input and trains a model that recovers the original uncorrupted input. Usually, the corrupted input are generated by setting some values of the input data record to default values generated based on a user-specified scheme (e.g., zeros or mean values on the corresponding attributes).

The dAE model seems like a nature fit for the MV imputation problem, as it aims to restore the corrupted values (i.e., MVs) by learning some intrinsic properties in uncorrupted values (i.e., non-missing values). However, simply applying a dAE model for MV imputation is impractical. First of all, a dAE model is originally designed to learn a good representation of the data, rather than MV imputation. Second, the objective function of dAE only considers to recover the entire original input rather than the MVs only, which leads to suboptimal imputation results. Finally, the corruption process in dAE is only introduced as a training criterion in order to obtain a better and more robust embedding instead of trying to recover values that are totally missing. As concluded by Bengio (Vincent et al. 2008), the dAE model is proposed to learn embeddings of the input that are robust to small irrelevent changes in input. In other words, the noises introduced in the corrupted input should not destruct the original data structure seriously. However, the MVs inherently existed in the real-world datasets may corrupt the stable structures and regular characteristics of the data, resulting in misguided learning of non-robust embeddings. Therefore, the performance of MV imputation by leveraging dAE directly is unsatisfactory (as demonstrated empirically in Sect. 4.3).

Motivated by the observed deficiencies of applying dAE to the MV imputation problem, we propose a dAE-based model of *MIssing Data Imputation denoising Autoencoder* (MIDIA, pronounced just like media), tailored for MV imputation. Given a dataset with MVs, MIDIA aims to capture the hidden correlations between MVs and non-MVs, and then estimates the MVs for imputation. Additionally, the proposed MIDIA is an MV-driven model, i.e., the model training processes and MV imputation strategies are different for various missing patterns. In this paper, we focus on three common missing patterns (McNeish 2017): *univariate missing pattern* (where the MVs occur only on a single attribute), *monotone missing pattern* (where the MVs concentrates on several attributes and the attributes can be sorted conveniently based on the percentage of missing values on each attribute) and *general missing pattern* (where the MVs may occur on any attribute). Accordingly, based on the MIDIA model, we devise two MV imputation approaches, namely MIDIA-Sequential and MIDIA-Batch, to accommodate data with various missing patterns. Among them, MIDIA-Sequential trains an individual MIDIA model for each *incomplete attribute* (attribute with MVs), and imputes the MVs on different incomplete attributes based on the corresponding learnt MIDIA models. Moreover, to further improve the imputation accuracy, MIDIA-Sequential imputes the MVs on different incomplete attributes sequentially. Similar to the sequential imputation strategy introduced in Zhang et al. (2008), the imputation starts from the incomplete attribute which has the least MVs and the imputed MVs are in turn used for imputing MVs on other incomplete attributes later. On the other hand, MIDIA-Batch trains a uniform MIDIA model and imputes the MVs in one batch. Moreover, with a rigid theoretical analysis (see Sect. 3.5 in detail) and extensive experiments, we find that MIDIA-Sequential and MIDIA-Batch can be reduced to the same approach for handling datasets with univariate missing pattern. MIDIA-Sequential is capable of handling datasets with monotone missing pattern, while MIDIA-Batch is capable of handling datasets with general missing pattern.

The major contributions made in this paper are summarized as follows.

- We analyze the pitfalls of MV imputation by simply using the dAE model. Accordingly, we propose a new dAE-based learning model, namely MIDIA, tailored for MV imputation.
- Considering three commonly discussed data missing patterns, we proposed two effective MV imputation approaches, namely MIDIA-Sequential and MIDIA-Batch, to exploit the proposed MIDIA model.
- We present an extensive experimental evaluation on real datasets. The results demonstrate that the proposed MIDIA and MV imputation approaches achieve significantly higher imputation accuracy than existing methods, and are competent to handle data with various data types and missing patterns.

The remainder of this paper is structured as follows. In Sect. 2, we introduce the preliminaries and related works. In Sect. 3, we present the methodology and then, in Sect. 4, we report the experiment results. Finally, we conclude the paper in Sect. 5.

2 Preliminaries

In this section, we first formulate the MV imputation problem. Next, we review some prior works relevant to our research. Finally, we provide some background on denoising autoencoder (dAE) which is the foundation of our research.

2.1 Problem formulation

We start by introducing some notation and definitions used in this paper and state our research goal.

Definition 1 An *observation* is a data record describing an object, which consist of d attributes.

Definition 2 An *incomplete observation* is a data record where the values on certain attributes are missing, while a *complete observation* is a data record where the values on all attributes exist.

Definition 3 Given a *dataset* consists of N observations, it can be represented by an $N \times d$ data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, where each vector $\mathbf{x}_i$ $(1 \leq i \leq N)$ is an observation, denoted by $\mathbf{x}_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,d}\right)$. Moreover, the dataset consists of two disjoint subsets: *incomplete dataset*, denoted by $\mathbf{X}_m$, and *complete dataset*, denoted by $\mathbf{X}_c$, containing incomplete observations and complete observations in $\mathbf{X}$, respectively.

Table 1 Example of data tuples with MVs

	CO	NMHC	NO_x	NO_2	O_3	T	RH	AH
X_1	0.088	0.016	0.491	0.079	0.046	0.135	0.415	0.084
X_2	?	0.017	0.488	0.052	0.031	0.114	0.374	0.060
X_3	?	0.019	0.512	?	0.040	0.129	?	0.096
X_4	0.037	0.022	0.569	0.158	0.051	0.273	0.525	0.231
X_5	?	0.022	0.534	?	0.036	0.159	0.418	0.099
X_6	0.096	0.023	0.545	0.052	0.017	0.123	0.488	0.099
X_7	?	0.023	0.615	0.114	0.035	0.265	0.370	0.147
X_8	0.076	0.023	0.528	0.108	0.050	0.189	0.504	0.150
X_9	0.022	0.024	0.614	0.136	0.028	0.245	0.541	0.213
X_{10}	?	0.025	0.612	0.120	0.033	0.271	0.377	0.156

Definition 4 Given a dataset $\mathbf{X}$, the *missing indicator matrix* is denoted by $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N]$ to indicate the MVs in $\mathbf{X}$, where the i-th vector $\mathbf{s}_i = \left(s_{i,1}, s_{i,2}, \ldots, s_{i,d}\right)$ is corresponding to the observation $\mathbf{x}_i$. If the value on the j-th attribute of an observation $\mathbf{x}_i$ is missing, then $s_{i,j} = 1$, otherwise $s_{i,j} = 0$.

Example 1 Table 1 shows a dataset (where each row is an observation) sampled from the AirQuality[1] dataset. The dataset contains various hourly averaged responses from an Air Quality Chemical Multi-sensor device deployed in a significantly polluted area of Italy (Vito et al. 2008). Each observation has eight attributes that are concentrations for CO, Non Metanic Hydrocarbons (NMHC), Total Nitrogen Oxides (NO_x), Nitrogen Dioxide (NO_2), Indium Oxide (O_3), Temperature (T), Relative Humidity (RH) and Absolute Humidity (AH). For illustration, we use '?' to indicate the MVs. As shown, the incomplete observations are $\{X_2, X_3, X_5, X_7, X_{10}\}$ (i.e., the values on some attributes of these observations are missing) and the complete observations are $\{X_1, X_4, X_6, X_8, X_9\}$.

As mentioned above, we aim to facilitate the MV imputation by exploring the non-linear correlations between MVs and non-MVs. Given a dataset $\mathbf{X}$, the core issue in this work is to design a proper structure of learning model tailored for MV imputaiton and return an effective imputed dataset $\mathbf{X}^*$ based on the proposed model by considering various missing patterns. To address the issues mentioned above, we propose a novel dAE-based learning model, namely MIDIA (*MIissing Data Imputation denoising Autoencoder*). Moreover, based on the proposed MIDIA model, we devise two effective MV imputation approaches, namely MIDIA-Sequential and MIDIA-Batch, for MV imputation of data with various missing patterns.

[1] https://archive.ics.uci.edu/ml/datasets/Air+Quality.

2.2 Related work

The existing MV imputation methods can be generally categorized into two classes (Magnani 2004): (i) *local imputation* which is based on the inter-correlations amongst observations and (ii) *global imputation* which is based on the intra-correlations amongst attributes in the same observation. The local imputation estimates the MV on an attribute of an observation based on the values on the same attribute of its neighbors. The global imputation estimates the MV on an attribute of an observation by modeling the correlations between the incomplete attribute and complete attributes in the same observation. Existing studies in the iterature show that these two classes of methods impute MVs from two orthogonal views and no one is absolutely better than the other. Different solutions fit for different situations.

The *k*NN-based imputation (Troyanskaya et al. 2001; Kim et al. 2004; Verboven et al. 2007; Zhang et al. 2007) which aims to find *k* nearest neighbors of an incomplete observation and then takes a distance-weighted mean of the *k* neighbors for imputation, is the most well-known local imputation method. In *k*NN imputation, the parameter *k* has a significant effect on the performance of the imputation. However, there is no theoretically optimal way to determine *k* properly and the *k* may be different for each dataset. Hot-deck imputation (Joenssen and Bankhofer 2012; Kim et al. 2005; Zhang et al. 2008) is another simple yet effective local imputation method. It partitions observations into disjoint groups and predicts the MVs by using values from one or more similar observations (donors) within the same group. Nevertheless, hot-deck-based procedures make a strong assumption that observations can be organized in classes with little variation inside a class. This contradicts from the assumption that data are thought as being independent and identically which is widely make in statistical methods and more likely to be true in real applications. Additionally, for both *k*NN-based and hot-deck-based imputation methods, the proper selection of similarity functions may be difficult especially for the data with heterogeneous attributes (i.e., the attributes are of different data types).

The global imputation methods leverage the correlations between the incomplete attribute and complete attributes in an observation itself to estimate the MVs. One such strategy is imputation by employing a regression model taking an observation as the input to predict the MVs using the non-missing values in the same observation. The model can be trained from complete observations in the data space based on EM (Expectation Maximization) algorithm (Dempster et al. 1977). Generally, linear regression (Wang and Rao 2002a, b; Yuan 2010) is a typical choice as the parameters of the linear regression model is easy to estimate. A few kernel-based MV imputation methods (Zhu et al. 2011; Qin et al. 2009) build a non-linear model to depict the non-linear correlations between incomplete attributes and complete attributes based on various kernel functions. However, it is notoriously hard to select a proper kernel function for the datasets with complex interactions and non-linear relation structures. Only recently, neural network models are developed for MV imputation since the neural networks have the capability to extract the complex correlations in the data. MIDA (Lovedeep and Wang 2017), an imputation model with deep denoising autoencoders, has been proposed. As MIDA imputes the MVs by directly applying dAE

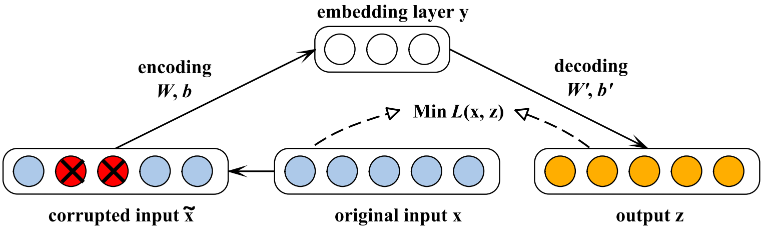


Fig. 1 Denoising autoencoder architecture

model, the improvement of imputation accuracy is limited (as introduced in Sect. 1). In this work, we propose a variant of dAE, named MIDIA, to tackle the problem of MV imputation, carry out a grid theoretical analysis and perform a holistic research on the problem.

2.3 Background—denoising autoencoder

In this section, we will briefly review the traditional denoising autoencoder to provide some background for our research.

A *denoising AutoEncoder* (dAE) (Vincent et al. 2010) is a powerful, non-linear mapping model to learn an effective representation with low dimensionality of the original data. Without loss of generality, in Fig. 1, we take a one-layer dAE model as an example for illustration.

First of all, to make the learnt model more robust and avoid overfitting, dAE corrupts the original input $\mathbf{x}$ into $\widetilde{\mathbf{x}}$ by adding some additive small noises (e.g., isotropic Gaussian noises) or forcing a fraction of elements in $\mathbf{x}$ to some default values (e.g., zeros or mean values on the corresponding attributes). In this paper, we focus on the later strategy for generating corrupted input because it can be viewed as removing part of elements in the original input and replacing their values by some default values which is a common technique for handling MVs. Next, the corrupted input $\widetilde{\mathbf{x}}$ is mapped to an h-dimensional hidden representation (embedding) $\mathbf{y} = f\left(\widetilde{\mathbf{x}}\mathbf{W} + \mathbf{b}\right)$ by an *encoder*, where $f\left(\cdot\right)$ is a user-specified activation function, $\mathbf{W}$ is a $d * h$ encoding weight matrix and $\mathbf{b}$ is a h encoding bias vector. Generally, the embedding layer has less dimensionality than the input, i.e., $h < d$, which corresponds to the regime where the dAE tries to implement data compression (Baldi 2012). Finally, the resulting embedding $\mathbf{y}$ is mapped back to reconstruct the original input $\mathbf{x}$ through a *decoder*. The transformation function has a similar formulation $\mathbf{z} = g\left(\mathbf{y}\mathbf{W}^{'} + \mathbf{b}^{'}\right)$, where $g\left(\cdot\right)$ is also a user-specified activation function, $\mathbf{W}^{'}$, $\mathbf{b}^{'}$ are the $h * d$ decoding weight matrix and d decoding bias vector respectively.

The objective function of dAE is to minimize the *reconstruction* error between the original input $\mathbf{x}$ and the output (reconstruction) $\mathbf{z}$, i.e., $\underset{\theta}{arg\ min} L\left(\mathbf{x}, \mathbf{z}\right)$ where $\theta = \{\mathbf{W}, \mathbf{W}^{'}, \mathbf{b}, \mathbf{b}^{'}\}$ is the parameter to be optimized and $L\left(\cdot\right)$ refers to a loss function to measure the distance between the input $\mathbf{x}$ and $\mathbf{z}$. It is notable that the the output $\mathbf{z}$ of dAE is a deterministic function of the corrupted input $\widetilde{\mathbf{x}}$ rather than the original input $\mathbf{x}$. While

$\mathbf{z}$ is expected to be as close as possible to the original input $\mathbf{x}$. The basic idea of the parameter optimization is that if the embedding $\mathbf{y}$ captures the useful features of the original input $\mathbf{x}$ from its corrupted version $\widetilde{\mathbf{x}}$, it allows a good reconstruction $\mathbf{z}$ of the original input $\mathbf{x}$. Therefore, by training the model to minimize reconstruction error amounts to generating a good embedding which retains much of the information in the original input.

3 Methodology

3.1 Exploring dAE to MV imputation

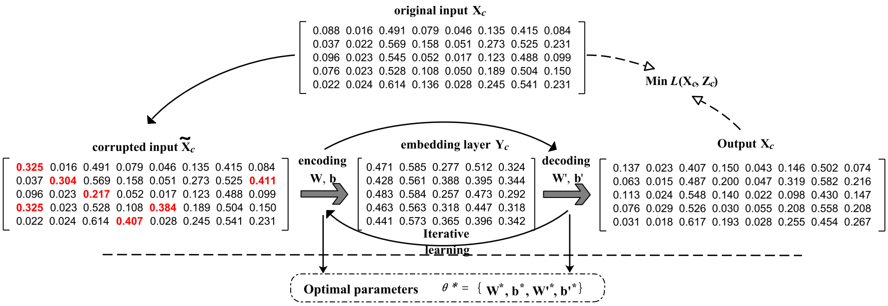
As introduced earlier, a dAE handles two things: (1) attempting to encode the corrupted input to obtain a good embedding and (2) attempting to undo the effect of a corruption process stochastically applied to the original input. In other words, besides good embedding learning, the dAE performs the *denoising* which can be used for MV imputation.

Given a dataset $\mathbf{X}$ with MVs, from the perspective of deep learning, we consider the complete dataset $\mathbf{X}_c \subseteq \mathbf{X}$ and the incomplete dataset $\mathbf{X}_m \subseteq \mathbf{X}$ as training set for model learning and testing set for MV imputation, respectively. For employing dAE to MV imputation, there are two phases: (1) *model training* trains a dAE model based on the complete dataset $\mathbf{X}_c$ and (2) *MV imputation* imputes the MVs in the incomplete dataset $\mathbf{X}_m$ based on the learned dAE model.

In model training phase, we take the complete dataset $\mathbf{X}_c$ as the original input. The corrupted input $\widehat{\mathbf{X}}_c$ is generated by randomly selecting some values of the original input $\mathbf{X}_c$ as synthetic MVs and replacing the ground truth of the MVs with default values (generated based on a user-specified scheme). Let the embedding and output (reconstruction) be $\mathbf{Y}_c$ and $\mathbf{Z}_c$, respectively. By minimizing the reconstruction error between $\mathbf{X}_c$ and its reconstruction $\mathbf{Z}_c$, the parameter θ is optimized (i.e., a dAE model is trained).

Example 2 Figure 2 shows the model training process over a training set consist of all complete observations in AirQuality dataset introduced in Example 1. In Fig. 2, we only show the complete observations in Table 1 due to the space limitation. For corrupted input generation, we randomly select some values in the original input as MVs, and replace the selected values (MVs) with the mean values of the corresponding attributes (where the mean values are considered as the default values). For example, the mean value of the first attribute CO in the dataset is 0.325, and thus we replace all synthetically generated MVs on attribute CO with 0.325. Unless noted specifically, we use the mean values of the corresponding attributes to replace the MVs in all examples through the paper. Based on the generated corrupted input, the parameter θ of the dAE model is optimized by iteratively learning the mapping functions in encoding and decoding.

Now given an incomplete dataset $\mathbf{X}_m$, we intend to impute the MVs in $\mathbf{X}_m$ using the learned dAE model. As the same line of model training, we first generate the corruption $\widetilde{\mathbf{X}}_m$ of $\mathbf{X}_m$ by replacing the MVs in $\mathbf{X}_m$ with the default values. Next, based


original input $\mathbf{X}_c$
Min $L(\mathbf{X}_c, \mathbf{Z}_c)$
corrupted input $\widetilde{\mathbf{X}}_c$
encoding
W, b
embedding layer $\mathbf{Y}_c$
decoding
W', b'
Output $\mathbf{X}_c$
Iterative learning
Optimal parameters $\theta^* = \{\mathbf{W}^*, \mathbf{b}^*, \mathbf{W}'^*, \mathbf{b}'^*\}$

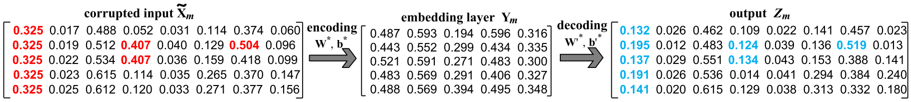
$$\mathbf{X}_c = \begin{bmatrix} 0.088 & 0.016 & 0.491 & 0.079 & 0.046 & 0.135 & 0.415 & 0.084 \\ 0.037 & 0.022 & 0.569 & 0.158 & 0.051 & 0.273 & 0.525 & 0.231 \\ 0.096 & 0.023 & 0.545 & 0.052 & 0.017 & 0.123 & 0.488 & 0.099 \\ 0.076 & 0.023 & 0.528 & 0.108 & 0.050 & 0.189 & 0.504 & 0.150 \\ 0.022 & 0.024 & 0.614 & 0.136 & 0.028 & 0.245 & 0.541 & 0.231 \end{bmatrix}$$

$$\widetilde{\mathbf{X}}_c = \begin{bmatrix} \mathbf{0.325} & 0.016 & 0.491 & 0.079 & 0.046 & 0.135 & 0.415 & 0.084 \\ 0.037 & \mathbf{0.304} & 0.569 & 0.158 & 0.051 & 0.273 & 0.525 & \mathbf{0.411} \\ 0.096 & 0.023 & \mathbf{0.217} & 0.052 & 0.017 & 0.123 & 0.488 & 0.099 \\ \mathbf{0.325} & 0.023 & 0.528 & 0.108 & \mathbf{0.384} & 0.189 & 0.504 & 0.150 \\ 0.022 & 0.024 & 0.614 & \mathbf{0.407} & 0.028 & 0.245 & 0.541 & 0.231 \end{bmatrix}$$

$$\mathbf{Y}_c = \begin{bmatrix} 0.471 & 0.585 & 0.277 & 0.512 & 0.324 \\ 0.428 & 0.561 & 0.388 & 0.395 & 0.344 \\ 0.483 & 0.584 & 0.257 & 0.473 & 0.292 \\ 0.463 & 0.563 & 0.318 & 0.447 & 0.318 \\ 0.441 & 0.573 & 0.365 & 0.396 & 0.342 \end{bmatrix}$$

$$\mathbf{X}_c = \begin{bmatrix} 0.137 & 0.023 & 0.407 & 0.150 & 0.043 & 0.146 & 0.502 & 0.074 \\ 0.063 & 0.015 & 0.487 & 0.200 & 0.047 & 0.319 & 0.582 & 0.216 \\ 0.113 & 0.024 & 0.548 & 0.140 & 0.022 & 0.098 & 0.430 & 0.147 \\ 0.076 & 0.029 & 0.526 & 0.030 & 0.055 & 0.208 & 0.558 & 0.208 \\ 0.031 & 0.018 & 0.617 & 0.193 & 0.028 & 0.255 & 0.454 & 0.267 \end{bmatrix}$$

Fig. 2 The model training of dAE


corrupted input $\widetilde{\mathbf{X}}_m$
encoding
$\mathbf{W}^*, \mathbf{b}^*$
embedding layer $\mathbf{Y}_m$
decoding
$\mathbf{W}'^*, \mathbf{b}'^*$
output Z_m

$$\widetilde{\mathbf{X}}_m = \begin{bmatrix} \mathbf{0.325} & 0.017 & 0.488 & 0.052 & 0.031 & 0.114 & 0.374 & 0.060 \\ \mathbf{0.325} & 0.019 & 0.512 & \mathbf{0.407} & 0.040 & 0.129 & \mathbf{0.504} & 0.096 \\ \mathbf{0.325} & 0.022 & 0.534 & \mathbf{0.407} & 0.036 & 0.159 & 0.418 & 0.099 \\ \mathbf{0.325} & 0.023 & 0.615 & 0.114 & 0.035 & 0.265 & 0.370 & 0.147 \\ \mathbf{0.325} & 0.025 & 0.612 & 0.120 & 0.033 & 0.271 & 0.377 & 0.156 \end{bmatrix}$$

$$\mathbf{Y}_m = \begin{bmatrix} 0.487 & 0.593 & 0.194 & 0.596 & 0.316 \\ 0.443 & 0.552 & 0.299 & 0.434 & 0.335 \\ 0.521 & 0.591 & 0.271 & 0.483 & 0.300 \\ 0.483 & 0.569 & 0.291 & 0.406 & 0.327 \\ 0.488 & 0.569 & 0.394 & 0.495 & 0.348 \end{bmatrix}$$

$$Z_m = \begin{bmatrix} \mathbf{0.132} & 0.026 & 0.462 & 0.109 & 0.022 & 0.141 & 0.457 & 0.023 \\ \mathbf{0.195} & 0.012 & 0.483 & \mathbf{0.124} & 0.039 & 0.136 & \mathbf{0.519} & 0.013 \\ \mathbf{0.137} & 0.029 & 0.551 & \mathbf{0.134} & 0.043 & 0.153 & 0.388 & 0.141 \\ \mathbf{0.191} & 0.026 & 0.536 & 0.014 & 0.041 & 0.294 & 0.384 & 0.240 \\ \mathbf{0.141} & 0.020 & 0.615 & 0.129 & 0.038 & 0.313 & 0.332 & 0.180 \end{bmatrix}$$

Fig. 3 The MVs imputation based on learned dAE

on the corrupted input $\widetilde{\mathbf{X}}_m$ and the optimized parameter θ^*, the reconstruction $\mathbf{Z}_m$ of the original input is output. Since in training phase, the output is the reconstruction of the complete dataset (the original input), the output $\mathbf{Z}_m$ in testing phase is also the reconstruction of the true version of $\mathbf{X}_m$ (MVs are replaced with ground truths). In other words, the values in $\mathbf{Z}_m$ corresponding to the MVs in $\mathbf{X}_m$ are the imputation results.

Example 3 For the incomplete dataset $\mathbf{X}_m$ in Example 1, the MV imputation process based on the learned dAE model in Example 2 is shown in Fig. 3. As shown, the MVs in $\mathbf{X}_m$ are first initialized by the mean values of the corresponding attributes (which is consistent with the data corruption process in model training phase). Then, the embedding $\mathbf{Y}_m$ and output $\mathbf{Z}_m$ are computed based on the optimized parameter θ^*, respectively. Finally, the imputation results are obtained from the output $\mathbf{Z}_m$.

Nevertheless, as discussed in the Introduction, the imputation results derived by simply applying a dAE model are unsatisfactory due to various reasons, which are demonstrated in the experimental evaluation (see Sect. 4 in detail). Therefore, we propose a novel dAE-based model, MIDIA (MIssing Data Imputation denoising Autoencoder), which primarily focuses on the goal of effectively imputing the MVs rather than re-constructing the original uncorrupted input. Moreover, we propose two MV imputation approaches based on the proposed MIDIA model, namely MIDIA-Sequential and MIDIA-Batch, to accommodate data with various missing patterns.

3.2 An overview

Figure 4 provides an overview of the system framework for the proposed MIDIA model and the MV imputation approaches. Given a dataset $\mathbf{X}$ with MVs as the input, it is divided into the incomplete dataset $\mathbf{X}_m$ and the complete dataset $\mathbf{X}_c$. The procedure of MV imputation consists of six steps as follows:

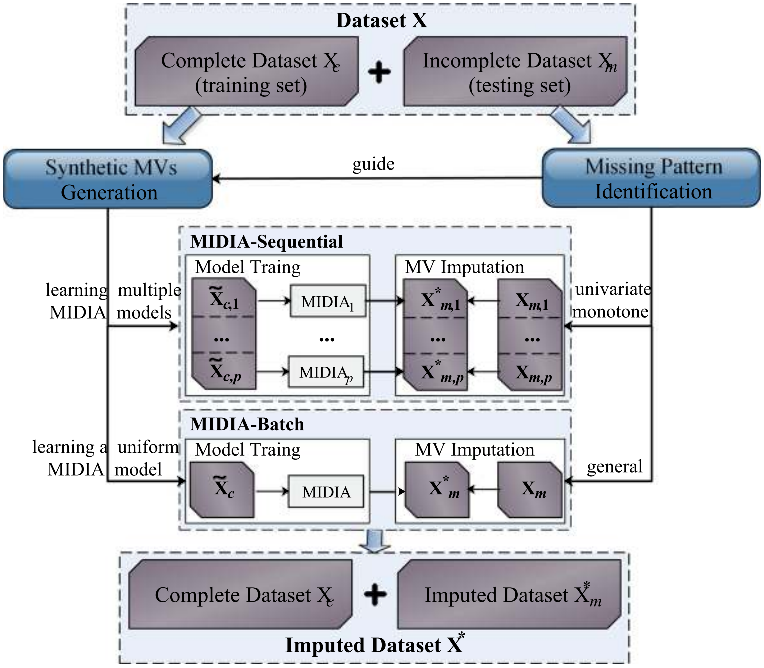


Fig. 4 System framework

Step 1 Identifying the missing pattern of the incomplete dataset $\mathbf{X}_m$ based on its missing indicator matrix $\mathbf{S}_m$.

Step 2 Introducing the synthetical MVs into the complete dataset X_c guided by the missing pattern of the incomplete dataset to generate the corrupted dataset $\widetilde{X}_c$ for training.

Step 3 Turning to step 4 if the missing pattern is univariate or monotone, otherwise turning to step 5.

Step 4 Dividing the incomplete dataset X_m into several subsets where each subset contains only one incomplete attribute. Then the MVs one each incomplete attribute are sequentially imputed based on MIDIA-Sequential.

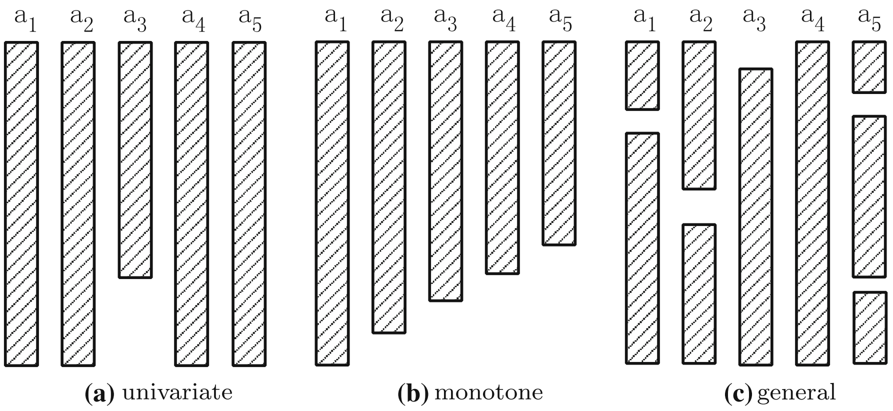
Step 5 Filling the MVs in one batch based on MIDIA-Batch.

Step 6 Returning the imputed dataset $\mathbf{X}^*$ finally.

Next, we introduce the missing pattern identification, the MIDIA model and the corresponding MV imputation approaches, i.e., MIDIA-Sequential and MIDIA-Batch, in detail respectively.

3.3 Missing pattern identification

As introduced earlier, the proposed MIDIA is an MV-driven model. For different missing patterns, the strategies for model training and MV imputation are different.



(a) univariate **(b)** monotone **(c)** general

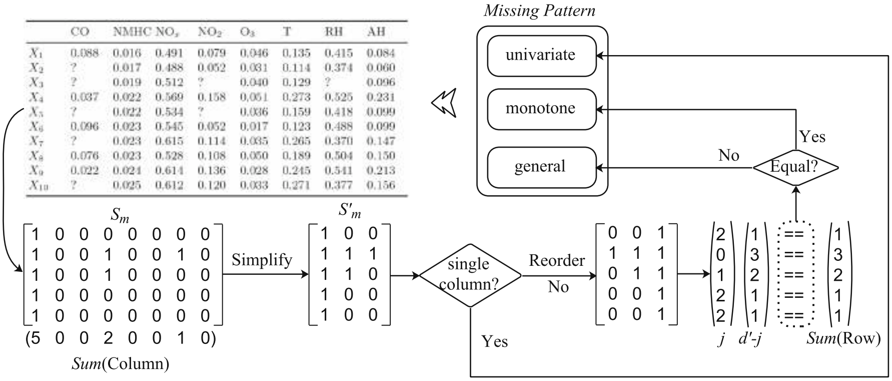
Fig. 5 Common missing patterns

Thus given an incomplete dataset X_m, we first introduce how to identify its missing pattern.

A missing pattern describes the arrangement of missing and non-missing values in the data (Jonathan et al. 2009). There are three missing patterns commonly discussed in the literature, i.e., univariate missing pattern, monotone missing pattern and general missing pattern, as illustrated in Fig. 5 where we assume there are five attributes $a_1 \sim a_5$ in an observation. With the univariate missing pattern, the MVs in the data appear only on a single attribute. As shown in Fig. 5a, the MVs only exist on the third attribute (i.e., a_3). With the monotone missing pattern, the MVs in the data appear on several attributes. Moreover, when the value on an attribute a_i of an observation is missing, all values on the subsequent attributes a_j $(j > i)$ of the same observation are also missing. As shown in Fig. 5b, when the value on attribute a_2 of an observation is missing, all values on $a_3 \sim a_5$ are also missing, i.e., the proportions of MVs on incomplete attributes are monotone. With the general missing pattern, the MVs may occur on any attribute randomly.

Specifically, given an incomplete dataset X_m, we determine its missing pattern based on the corresponding missing indicator matrix S_m. Based on Definition 4, when the value on the j-th attribute of the i-th observation $\mathbf{x}_i \in X_m$ is missing, $s_{ij} = 1$, otherwise $s_{ij} = 0$. Therefore, in matrix S_m, the sum of each row is the number of MVs in each observation, while the sum of each column is the number of MVs on each attribute. If the sum of a row (column) is zero, there is no MV in the observation (on the attribute). Since the complete attributes (i.e., the attributes without MVs) do not affect the missing pattern identification, we remove them (the columns where the sums are zero) from the missing indicator matrix S_m, and let $S_m^{'}$ be the simplified missing indicator matrix with d' incomplete attributes. The univariate missing pattern is easy to determine by examining if there is only one column (attribute) remaining in $S_m^{'}$. Next, we introduce how to determine if the missing pattern of X_m is monotone.

We first reorder the incomplete attributes in $S_m^{'}$ based on the number of MVs on each attribute in ascending order. Afterwards, for each row in $S_m^{'}$, when the first ‘1’ (i.e., the first MV) appears, all the values on its later attributes are ‘1’ under the scenario of the



	CO	NMHC	NO$_x$	NO$_2$	O$_3$	T	RH	AH
X_1	0.088	0.016	0.491	0.079	0.046	0.135	0.415	0.084
X_2	?	0.017	0.488	0.052	0.031	0.114	0.374	0.060
X_3	?	0.019	0.512	?	0.040	0.129	?	0.096
X_4	0.037	0.022	0.569	0.158	0.051	0.273	0.525	0.231
X_5	?	0.022	0.534	?	0.036	0.159	0.418	0.099
X_6	0.096	0.023	0.545	0.052	0.017	0.123	0.488	0.099
X_7	?	0.023	0.615	0.114	0.035	0.265	0.370	0.147
X_8	0.076	0.023	0.528	0.108	0.050	0.189	0.504	0.150
X_9	0.022	0.024	0.614	0.136	0.028	0.245	0.541	0.213
X_{10}	?	0.025	0.612	0.120	0.033	0.271	0.377	0.156

Fig. 6 Missing pattern identification

monotone missing pattern. The criterion is determined based on the definition of the monotone missing pattern introduced earlier, i.e., when the value on an attribute of an observation is missing, all values on the subsequent attributes of the same observation are also missing. Specifically, for the i-th row in $S_m^{'}$, suppose the index of the first '1' is j (which starts from 0), then the number of '1' in this row should be $d' - j$, i.e., the sum of the i-th row in $S_m^{'}$ should equal to $d' - j$. If all rows in $S_m^{'}$ satisfy the above criterion, the missing pattern of the incomplete dataset X_m is monotone; otherwise, the missing pattern of the incomplete dataset X_m is general.

Example 4 For the incomplete dataset $\mathbf{X}_m$ in Example 1, the process of missing pattern identification is shown in Fig. 6. First, the simplified missing indicator matrix $S_m^{'}$ (with three incomplete attributes, i.e., $d' = 3$) is derived by removing the complete attributes in the original missing indicator matrix S_m. Next, it is obvious that the missing pattern of $\mathbf{X}_m$ is not univariate since there are three incomplete attributes (columns) remaining in $S_m^{'}$. Then we reorder the incomplete attributes in $S_m^{'}$ based on the number of MVs on each incomplete attribute in ascending order. As shown, the numbers of MVs on the three incomplete attributes are 5, 2, 1, respectively. Thus after reordering, $S_m^{'}$ is flipped. Afterwards, we find the indexes of first '1' appearing in each row (i.e., $j = (2, 0, 1, 2, 2)^T$) and compute the results of $d' - j = (1, 3, 2, 1, 1)^T$. Finally, if the sum of each row (i.e., $(1, 3, 2, 1, 1)^T$) equals to the corresponding $d' - j$, the missing pattern of $\mathbf{X}_m$ is monotone, otherwise the missing pattern is general.

3.4 MIDIA model

We take a one-layer MIDIA model as an example for illustration, shown in Fig. 7 where the red part in each node of corrupted input represents the synthetically generated MVs and the yellow part in each node of the output layer is the corresponding reconstructions.

Given an original input $\mathbf{x}$, the data transformation between each layer in MIDIA is described as follows.

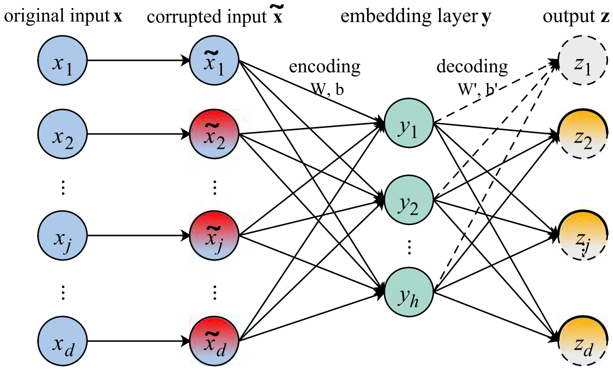


Fig. 7 The architecture of MIDIA model

(1) *Corrupted input generation*: MIDIA first generates the corrupted input $\widetilde{\mathbf{x}}$ by marking off some values (i.e., synthetic MVs) of the original input $\mathbf{x}$ and filling the sythetic MVs with some default values. Note that to accommodate effective MV imputation, the missing pattern of synthetically generated corrupted input in the training set needs to be consistent with that of the incomplete dataset to be imputed. We discuss this issue later.

(2) *Encoding*: An *encoder* transforms the corrupted input $\widetilde{\mathbf{x}}$ into an h-dimensional embedding $\mathbf{y}$. Figure 7 shows a simple full-connected layer as the encoder for illustration, i.e., $\mathbf{y} = f\left(\mathbf{W}\widetilde{x} + \mathbf{b}\right)$.

(3) *Decoding*: A decoder takes the embedding $\mathbf{y}$ learned by the encoder as the input and transforms it back to $\mathbf{z}$ which aims to reconstruct the original input $\mathbf{x}$. Again, Fig. 7 shows a simple fully-connected layer as the decoder for illustration, i.e., $\mathbf{z} = g\left(\mathbf{W}^{'}\mathbf{y} + \mathbf{b}^{'}\right)$.

In the encoding and decoding steps, the encoder $f(\cdot)$ and decoder $g(\cdot)$ are non-linear activation functions to generate the embedding $\mathbf{y}$ and the reconstruction $\mathbf{z}$, respectively. Various non-linear activation functions have been proposed in the literature, e.g., Sigmoid (Han and Moraga 1995), TanH (Sinclair et al. 2001), Softsign (Bergstra et al. 2009; Glorot and Bengio 2010), SoftPlus (Glorot et al. 2011), ReLU (Nair and Hinton 2010). Since different activation functions fit for different data and situations, we adopt different alternative activation functions during the MV imputation based on MIDIA and evaluate the their performance in Sect. 4.

Although the network architecture in MIDIA looks the same as the dAE model, MIDIA cares more about the reconstruction accuracy of those MVs since the goal of MIDIA is MV imputation. To achieve this goal, we design the objective function of MIDIA to minimize the *reconstruction error of MVs* rather than the reconstruction error of the whole input observations, as derived in Eq. (1) below.

$$arg\,\underset{\theta}{min}\frac{1}{m}\sum_{i=1}^{n} L\left(\mathbf{s}_i \cdot \mathbf{x}_i, \mathbf{s}_i \cdot \mathbf{z}_i\right) \tag{1}$$

where $\mathbf{x}_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,d}\right) \in \mathbf{X}_t$ is an original observation (uncorrupted) in the training set, and $\mathbf{z}_i = \left(z_{i,1}, z_{i,2}, \ldots, z_{i,d}\right) \in \mathbf{Z}_t$ is the reconstruction of $\mathbf{x}_i$ in the output. The missing indicator vector $\mathbf{s}_i = \left(s_{i,1}, s_{i,2}, \ldots, s_{i,d}\right)$ corresponding to $\mathbf{x}_i$ indicates the MVs occurrence in $\mathbf{x}_i$, where $s_{i,j} = 1$ if $x_{i,j}$ is a missing value, otherwise $s_{i,j} = 0$. The $\mathbf{s}_i \cdot \mathbf{x}_i$ and $\mathbf{s}_i \cdot \mathbf{z}_i$ compute the inner product of $(\mathbf{s}_i, \mathbf{x}_i)$ and $(\mathbf{s}_i, \mathbf{z}_i)$, respectively. Finally, the parameter $\theta = \{\mathbf{W}, \mathbf{W}^{'}, \mathbf{b}, \mathbf{b}^{'}\}$ is initialized at random, and optimized by stochastic gradient descent (Bottou 2010). It is notable that the loss function can be tailored for different data types. For numerical data, we adopt the *square error* loss function $L(\cdot)$ [as shown in Eq. (2)], while for categorical data, we adopt the *cross-entropy* loss function [as shown in Eq. (3)] with one-hot encoding. Moreover, for the mixed types of data, above two loss functions are weighted unified together to generate the final loss function [as shown in Eq. (4)].

$$L\left(\mathbf{x}_i^{'}, \mathbf{z}_i^{'}\right) = \sum_{1 \leq j \leq d} \left(x_{i,j}^{'} - z_{i,j}^{'}\right) \tag{2}$$

$$L\left(\mathbf{x}_i^{'}, \mathbf{z}_i^{'}\right) = -\sum_{1 \leq j \leq d} \left[x_{i,j}^{'} log z_{i,j}^{'} + (1 - x_{i,j}^{'}) log(1 - z_{i,j}^{'})\right] \tag{3}$$

$$L\left(\mathbf{x}_i^{'}, \mathbf{z}_i^{'}\right) = w_n \sum_{1 \leq j \leq d_n} \left(x_{i,j}^{'} - z_{i,j}^{'}\right) - w_c \sum_{d_n < j \leq d} \left[x_{i,j}^{'} log z_{i,j}^{'} + (1 - x_{i,j}^{'}) log(1 - z_{i,j}^{'})\right] \tag{4}$$

where $\mathbf{x}_i^{'} = \mathbf{s}_i \cdot \mathbf{x}_i$, $\mathbf{z}_i^{'} = \mathbf{s}_i \cdot \mathbf{z}_i$ and d_n is the number of numerical attributes in the mixed type of data. In addition, w_n and w_c are the weights of numerical attributes and categorical attributes, respectively, and we have $w_n + w_c = 1$.

Based on the above introduction, compared with traditional dAE model, there are mainly two modifications in the proposed MIDIA model. First, the corrupted input in MIDIA model is generated with the guidance of the missing pattern in the incomplete dataset to be imputed. Second, the objective function of MIDIA model is to minimize the reconstruction error of MVs rather than that of the whole input in dAE. The rationale behind the modifications is two-fold, as illustrated below.

(1) The machine learning models we aim to construct is data-dependent, which contains two intuitive meanings: (i) *The training set and testing set are similar to each other on the data content.* For example, a dAE model trained based on a dataset (training set) with images about trees does not perform well in compressing a dataset (testing set) with images about dogs. The reason is that the features learned in the model training are about trees which obviously cannot describe dogs accurately. (ii) *The distributions of training set and testing set are close to each other.* As introduced in Borovicka et al. (2012), for a reliable future error prediction, we need to evaluate our model on a different, independent and **identically distributed** (testing) set that is different to the (training) set we have used for building the model.

Similar to the intuitions of data-dependency in machine learning, the proposed MIDIA model in this paper serves to fill MVs in a given incomplete dataset (i.e., testing set). Thus the MVs in the training set and testing set are assumed to follow the same distribution (with a specific tolerance of deviation), especially when the missing rate is relatively high.

(2) As introduced earlier, the proposed MIDIA aims to recover the MVs accurately. To achieve this goal, MIDIA is forced to extract correlations between the MVs and non-MVs for effective MV imputation by only caring about the reconstruction accuracy of those MVs.

3.5 MV imputation based on MIDIA

Given a dataset $\mathbf{X}$ with MVs, our goal is to impute the MVs in the incomplete dataset $\mathbf{X}_m \subseteq \mathbf{X}$ effectively. To achieve this goal, as the same line of employing dAE to MV imputation, there are two phases, i.e., *model training* to learn an effective MIDIA model based on the complete dataset $\mathbf{X}_c$ and *MV imputation* to fill the MVs in $\mathbf{X}_m$ based on the learnt MIDIA model.

In model training phase, as introduced in Sect. 3.4, the corrupted input generation in MIDIA depends on the missing pattern of the incomplete dataset to be imputed. A missing pattern describes the arrangement of missing and non-missing values in the data (Jonathan et al. 2009). There are three missing patterns commonly discussed in the literature, i.e., univariate missing pattern, monotone missing pattern and general missing pattern, as illustrated in Fig. 5 where we assume there are five attributes $a_1 \sim a_5$ in an observation. With the univariate missing pattern, the MVs in the data appear only on a single attribute. As shown in Fig. 5a, the MVs only exist on the third attribute (i.e., a_3). With the monotone missing pattern, the MVs in the data appear on several attributes. Moreover, when the value on an attribute a_i of an observation is missing, all values on the subsequent attributes a_j $(j > i)$ of the same observation are also missing. As shown in Fig. 5b, when the value on attribute a_2 of an observation is missing, all values on $a_3 \sim a_5$ are also missing, i.e., the proportions of MVs on each incomplete attributes are monotone. With the general missing pattern, the MVs may occur on any attribute. For different missing patterns, the imputation strategies are different. Therefore, we propose two MV imputation approaches based on the MIDIA model to adapt various missing data patterns.

3.5.1 MIDIA-sequential

The basic idea behind this approach is to impute the MVs on each incomplete attribute independently and sequentially.

Given a dataset $\mathbf{X}$, suppose there are p incomplete attributes and $d - p$ complete attributes in an observation. For each incomplete attribute a_i $(1 \leq i \leq p)$, we aim to impute the MVs on a_i using the observed values on complete attributes by training a MIDIA model. Moreover, once the MVs on an incomplete attribute a_i are imputed, a_i is considered as a complete attribute and used for imputing the MVs on other incomplete attribute later. To alleviate the effect of inaccuracy in the imputed values, we start with sequential imputation from the incomplete attribute which has the least MVs. For example, there are three incomplete attributes CO, NO_2 and RH in the data shown in Table 1. The number of MVs on each incomplete attribute is 5, 2 and 1, respectively. Thus the imputation is performed sequentially on RH, NO_2 and finally CO.

$$\mathbf{X}_{m,1}\,(\mathbf{RH})$$
$$\begin{array}{cccccc} \text{NMHC} & \text{NO}_x & \text{O}_3 & \text{T} & \text{RH} & \text{AH} \\ \hline 0.019 & 0.512 & 0.040 & 0.129 & ? & 0.096 \end{array}$$

$$\mathbf{X}_{m,2}\,(\mathbf{NO_2})$$
$$\begin{array}{ccccccc} \text{NMHC} & \text{NO}_x & \text{NO}_2 & \text{O}_3 & \text{T} & \text{RH} & \text{AH} \\ \hline 0.019 & 0.512 & ? & 0.040 & 0.129 & 0.504 & 0.096 \\ 0.022 & 0.534 & ? & 0.036 & 0.159 & 0.418 & 0.099 \end{array}$$

$$\mathbf{X}_{m,3}\,(\mathbf{CO})$$
$$\begin{array}{cccccccc} \text{CO} & \text{NMHC} & \text{NO}_x & \text{NO}_2 & \text{O}_3 & \text{T} & \text{RH} & \text{AH} \\ \hline ? & 0.017 & 0.488 & 0.052 & 0.031 & 0.114 & 0.374 & 0.060 \\ ? & 0.019 & 0.512 & 0.031 & 0.040 & 0.129 & 0.374 & 0.096 \\ ? & 0.022 & 0.534 & 0.031 & 0.036 & 0.159 & 0.418 & 0.099 \\ ? & 0.023 & 0.615 & 0.114 & 0.035 & 0.265 & 0.370 & 0.147 \\ ? & 0.025 & 0.612 & 0.120 & 0.033 & 0.271 & 0.377 & 0.156 \end{array}$$

Fig. 8 Subsets of incomplete dataset $\mathbf{X}_m$ in Example 1

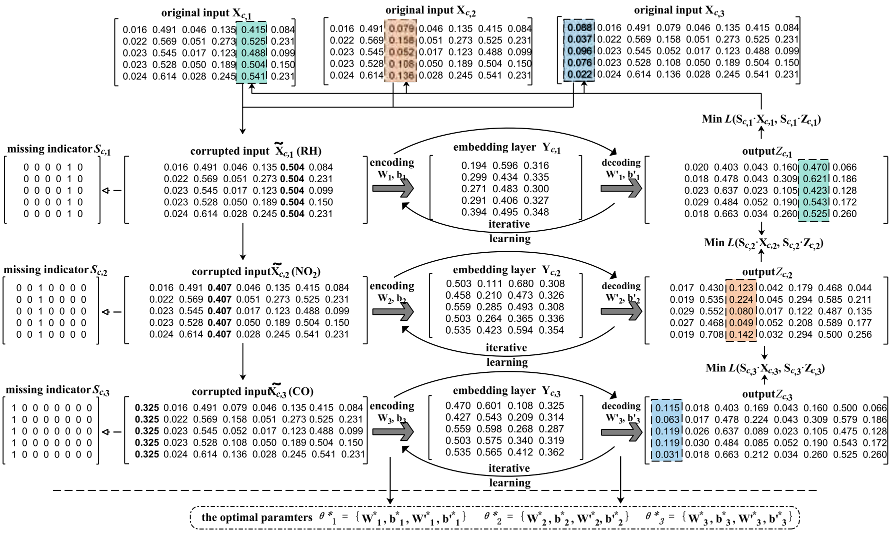
It is notable that, in MIDIA-Sequential, we train a MIDIA model for each incomplete attribute and impute the MVs on different incomplete attributes sequentially. Given an incomplete dataset $\mathbf{X}_m$, corresponding to each incomplete attribute, there is an incomplete subset $\mathbf{X}_{m,i} \subseteq \mathbf{X}_m$ $(1 \leq i \leq p)$ which consists of the observations with values on the i-th incomplete attribute missing. Moreover, the observations in $\mathbf{X}_{m,i}$ only contains the i-th incomplete attributes and all complete attributes, i.e., the incomplete attributes except for the i-th attribute are discarded in the training data preparation.

Example 5 For the incomplete dataset in Example 1, as introduced earlier, there are three incomplete attributes and the imputation is performed on attributes RH, NO_2 and CO sequentially. Thus we partition the incomplete dataset $\mathbf{X}_m$ into three incomplete subsets as shown in Fig. 8. Each subset represents a target incomplete dataset to be imputed where MVs (marked by '?' in Fig. 8) only occur on a single attribute. For example, the subset $\mathbf{X}_{m,2}$, corresponding to the second incomplete attribute NO_2 to be imputed, contains values from the incomplete attribute NO_2 and all complete attributes NMHC, NO_x, O_3, T, RH and AH. Note here that RH is considered as a complete attribute since the MVs on RH have been imputed.

For each of the incomplete dataset, MIDIA-Sequential trains a specific MIDIA model and employs it to impute the MVs on the corresponding incomplete attribute.

Model Training For an incomplete subset $\mathbf{X}_{m,i}$ $(1 \leq i \leq p)$, the model training phase takes the complete dataset $\mathbf{X}_{c,i} \subseteq \mathbf{X}_c$ which contains the same attributes with $\mathbf{X}_{m,i}$ as the input. As introduced earlier, to make the learned model fit for effective imputation of MVs in the target dataset (i.e., the incomplete subset $\mathbf{X}_{m,i}$), the missing pattern of synthetically generated MVs in the corrupted input $\widetilde{\mathbf{X}}_{c,i}$ should be consistent with that in $\mathbf{X}_{m,i}$. Since the incomplete subset $\mathbf{X}_{m,i}$ only contains one incomplete attribute, and all values on the incomplete attribute are missing, i.e., the missing pattern of $\mathbf{X}_{m,i}$ is univariate, we generate the corrupted input $\widetilde{\mathbf{X}}_{c,i}$ by deleting all the observed values on the i-th incomplete attribute and replacing them with default values (generated based on a user-specified scheme). Next, based on the generated corrupted input, a MIDIA model used for imputing MVs in the incomplete subset $\mathbf{X}_{m,i}$ is trained.

Example 6 The model training process of MIDIA for the incomplete dataset $\mathbf{X}_m$ in Example 1 is shown in Fig. 9. By minimizing the reconstruction error between the reconstructed MVs in $\mathbf{Z}_{c,1}$ and the ground truth in $\mathbf{X}_{c,1}$, a MIDIA model for imputing the MVs on the incomplete attribute RH is trained. In the same way, the MIDIA models for the incomplete subset $\mathbf{X}_{m,2}$ (where the incomplete attribute is NO_2) and $\mathbf{X}_{m,3}$ (where the incomplete attribute is CO) are trained, respectively.



original input $X_{c,1}$
original input $X_{c,2}$
original input $X_{c,3}$
Min $L(S_{c,1}\cdot X_{c,1}, S_{c,1}\cdot Z_{c,1})$
missing indicator $S_{c,1}$
corrupted input $\widetilde{X}_{c,1}$ (RH)
encoding W_1, b_1
embedding layer $Y_{c,1}$
decoding W'_1, b'_1
output $Z_{c,1}$
iterative learning
Min $L(S_{c,2}\cdot X_{c,2}, S_{c,2}\cdot Z_{c,2})$
missing indicator $S_{c,2}$
corrupted input $\widetilde{X}_{c,2}$ (NO_2)
encoding W_2, b_2
embedding layer $Y_{c,2}$
decoding W'_2, b'_2
output $Z_{c,2}$
iterative learning
Min $L(S_{c,3}\cdot X_{c,3}, S_{c,3}\cdot Z_{c,3})$
missing indicator $S_{c,3}$
corrupted input $\widetilde{X}_{c,3}$ (CO)
encoding W_3, b_3
embedding layer $Y_{c,3}$
decoding W'_3, b'_3
output $Z_{c,3}$
iterative learning
the optimal paramters $\theta^*_1 = \{W^*_1, b^*_1, W'^*_1, b'^*_1\}$ $\theta^*_2 = \{W^*_2, b^*_2, W'^*_2, b'^*_2\}$ $\theta^*_3 = \{W^*_3, b^*_3, W'^*_3, b'^*_3\}$

original input $X_{c,1}$

$$\begin{bmatrix} 0.016 & 0.491 & 0.046 & 0.135 & 0.415 & 0.084 \\ 0.022 & 0.569 & 0.051 & 0.273 & 0.525 & 0.231 \\ 0.023 & 0.545 & 0.017 & 0.123 & 0.488 & 0.099 \\ 0.023 & 0.528 & 0.050 & 0.189 & 0.504 & 0.150 \\ 0.024 & 0.614 & 0.028 & 0.245 & 0.541 & 0.231 \end{bmatrix}$$

original input $X_{c,2}$

$$\begin{bmatrix} 0.016 & 0.491 & 0.079 & 0.046 & 0.135 & 0.415 & 0.084 \\ 0.022 & 0.569 & 0.158 & 0.051 & 0.273 & 0.525 & 0.231 \\ 0.023 & 0.545 & 0.052 & 0.017 & 0.123 & 0.488 & 0.099 \\ 0.023 & 0.528 & 0.108 & 0.050 & 0.189 & 0.504 & 0.150 \\ 0.024 & 0.614 & 0.136 & 0.028 & 0.245 & 0.541 & 0.231 \end{bmatrix}$$

original input $X_{c,3}$

$$\begin{bmatrix} \mathbf{0.088} & 0.016 & 0.491 & 0.079 & 0.046 & 0.135 & 0.415 & 0.084 \\ \mathbf{0.037} & 0.022 & 0.569 & 0.158 & 0.051 & 0.273 & 0.525 & 0.231 \\ \mathbf{0.096} & 0.023 & 0.545 & 0.052 & 0.017 & 0.123 & 0.488 & 0.099 \\ \mathbf{0.076} & 0.023 & 0.528 & 0.108 & 0.050 & 0.189 & 0.504 & 0.150 \\ \mathbf{0.022} & 0.024 & 0.614 & 0.136 & 0.028 & 0.245 & 0.541 & 0.231 \end{bmatrix}$$

missing indicator $S_{c,1}$

$$\begin{bmatrix} 0&0&0&0&1&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&1&0 \end{bmatrix}$$

corrupted input $\widetilde{X}_{c,1}$ (RH)

$$\begin{bmatrix} 0.016 & 0.491 & 0.046 & 0.135 & \mathbf{0.504} & 0.084 \\ 0.022 & 0.569 & 0.051 & 0.273 & \mathbf{0.504} & 0.231 \\ 0.023 & 0.545 & 0.017 & 0.123 & \mathbf{0.504} & 0.099 \\ 0.023 & 0.528 & 0.050 & 0.189 & \mathbf{0.504} & 0.150 \\ 0.024 & 0.614 & 0.028 & 0.245 & \mathbf{0.504} & 0.231 \end{bmatrix}$$

embedding layer $Y_{c,1}$

$$\begin{bmatrix} 0.194 & 0.596 & 0.316 \\ 0.299 & 0.434 & 0.335 \\ 0.271 & 0.483 & 0.300 \\ 0.291 & 0.406 & 0.327 \\ 0.394 & 0.495 & 0.348 \end{bmatrix}$$

output $Z_{c,1}$

$$\begin{bmatrix} 0.020 & 0.403 & 0.043 & 0.160 & 0.470 & 0.066 \\ 0.018 & 0.478 & 0.043 & 0.309 & 0.621 & 0.186 \\ 0.023 & 0.637 & 0.023 & 0.105 & 0.423 & 0.128 \\ 0.029 & 0.484 & 0.052 & 0.190 & 0.543 & 0.172 \\ 0.018 & 0.663 & 0.034 & 0.260 & 0.525 & 0.260 \end{bmatrix}$$

missing indicator $S_{c,2}$

$$\begin{bmatrix} 0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0 \end{bmatrix}$$

corrupted input $\widetilde{X}_{c,2}$ (NO_2)

$$\begin{bmatrix} 0.016 & 0.491 & \mathbf{0.407} & 0.046 & 0.135 & 0.415 & 0.084 \\ 0.022 & 0.569 & \mathbf{0.407} & 0.051 & 0.273 & 0.525 & 0.231 \\ 0.023 & 0.545 & \mathbf{0.407} & 0.017 & 0.123 & 0.488 & 0.099 \\ 0.023 & 0.528 & \mathbf{0.407} & 0.050 & 0.189 & 0.504 & 0.150 \\ 0.024 & 0.614 & \mathbf{0.407} & 0.028 & 0.245 & 0.541 & 0.231 \end{bmatrix}$$

embedding layer $Y_{c,2}$

$$\begin{bmatrix} 0.503 & 0.111 & 0.680 & 0.308 \\ 0.458 & 0.210 & 0.473 & 0.326 \\ 0.559 & 0.285 & 0.493 & 0.308 \\ 0.503 & 0.264 & 0.365 & 0.336 \\ 0.535 & 0.423 & 0.594 & 0.354 \end{bmatrix}$$

output $Z_{c,2}$

$$\begin{bmatrix} 0.017 & 0.430 & 0.123 & 0.042 & 0.179 & 0.468 & 0.044 \\ 0.019 & 0.535 & 0.224 & 0.045 & 0.294 & 0.585 & 0.211 \\ 0.029 & 0.552 & 0.080 & 0.017 & 0.122 & 0.487 & 0.135 \\ 0.027 & 0.468 & 0.049 & 0.052 & 0.208 & 0.589 & 0.177 \\ 0.019 & 0.708 & 0.142 & 0.032 & 0.294 & 0.500 & 0.256 \end{bmatrix}$$

missing indicator $S_{c,3}$

$$\begin{bmatrix} 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \end{bmatrix}$$

corrupted input $\widetilde{X}_{c,3}$ (CO)

$$\begin{bmatrix} \mathbf{0.325} & 0.016 & 0.491 & 0.079 & 0.046 & 0.135 & 0.415 & 0.084 \\ \mathbf{0.325} & 0.022 & 0.569 & 0.158 & 0.051 & 0.273 & 0.525 & 0.231 \\ \mathbf{0.325} & 0.023 & 0.545 & 0.052 & 0.017 & 0.123 & 0.488 & 0.099 \\ \mathbf{0.325} & 0.023 & 0.528 & 0.108 & 0.050 & 0.189 & 0.504 & 0.150 \\ \mathbf{0.325} & 0.024 & 0.614 & 0.136 & 0.028 & 0.245 & 0.541 & 0.231 \end{bmatrix}$$

embedding layer $Y_{c,3}$

$$\begin{bmatrix} 0.470 & 0.601 & 0.108 & 0.325 \\ 0.427 & 0.543 & 0.209 & 0.314 \\ 0.559 & 0.598 & 0.268 & 0.287 \\ 0.503 & 0.575 & 0.340 & 0.319 \\ 0.535 & 0.565 & 0.412 & 0.362 \end{bmatrix}$$

output $Z_{c,3}$

$$\begin{bmatrix} 0.115 & 0.018 & 0.403 & 0.169 & 0.043 & 0.160 & 0.500 & 0.066 \\ 0.063 & 0.017 & 0.478 & 0.224 & 0.043 & 0.309 & 0.579 & 0.186 \\ 0.119 & 0.026 & 0.637 & 0.089 & 0.023 & 0.105 & 0.475 & 0.128 \\ 0.119 & 0.030 & 0.484 & 0.085 & 0.052 & 0.190 & 0.543 & 0.172 \\ 0.031 & 0.018 & 0.663 & 0.212 & 0.034 & 0.260 & 0.525 & 0.260 \end{bmatrix}$$

Fig. 9 the model training of MIDIA-Sequential approach

MV Imputation In this phase, we impute the MVs in each incomplete subset $\mathbf{X}_{m,i}$ (where the MVs only occur on the i-th incomplete attribute) by using its corresponding trained MIDIA models. As mentioned ealier, the imputation starts from the the incomplete attribute having the least MVs, i.e., the incomplete subset which has the fewest observations, and the imputed MVs are used for later imputation. For the i-th incomplete attribute, we first initialize the MVs in $\mathbf{X}_{m,i}$ by default values adopted in the training phase. Notice that we only regard MVs on the i-th incomplete attribute as the MVs to be imputed, the MVs imputed previously on other incomplete attributes (which are taken as the complete attributes when conducting MV imputation for the i-th incomplete attribute) are taken as the "ground truth". Taking the initialized incomplete subset $\mathbf{X}_{m,i}$ as the corrupted input, through the mapping functions in encoding an decoding, the imputation results of MVs in $\mathbf{X}_{m,i}$ can be found in the reconstruction $\mathbf{Z}_{m,i}$. After the MVs in all incomplete subsets being imputed sequentially, the final imputed dataset $\mathbf{X}^*$ is thus derived.

Example 7 Based on the MIDIA-Sequential approach, the MV imputation for the incomplete dataset $\mathbf{X}_m$ in Example 1 is shown in Fig. 10. First, the subset $\mathbf{X}_{m,1}$, i.e., the MVs on the incomplete attribute RH is imputed, as it contains the fewest MVs. Additionally, the imputed values are in turn used for later imputation, i.e., in the corrupted input $\widetilde{\mathbf{X}}_{m,2}$ and $\widetilde{\mathbf{X}}_{m,3}$, the MVs on attribute RH are replaced with the imputed values. Then the subsets $\mathbf{X}_{m,2}$ and $\mathbf{X}_{m,3}$ which have two and five MVs are imputed sequentially.

3.5.2 MIDIA-Batch

The basic idea behind this approach is to impute the MVs in the incomplete dataset $\mathbf{X}_m$ in one batch by training a uniform MIDIA model.

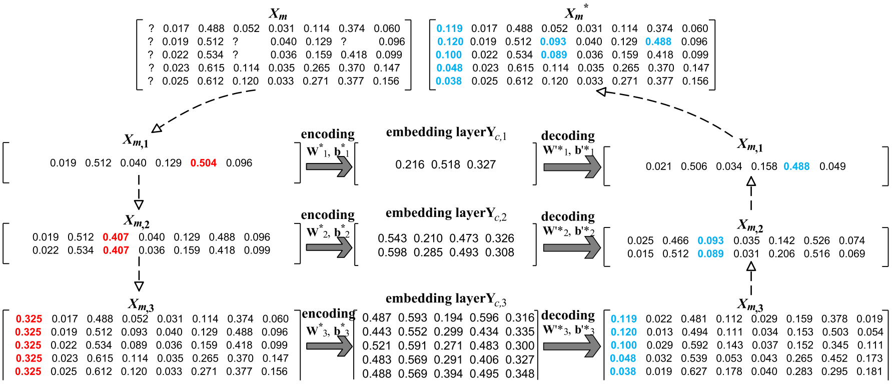


Fig. 10 MV imputation process based on MIDIA-Sequential approach

In model training phase, taking the complete dataset $\mathbf{X}_c$ as the input, the first step is to generate a corrupted input $\widetilde{\mathbf{X}}_c$ by selecting a fraction of elements in $\mathbf{X}_c$ as MVs and replacing them with some default values. To make the missing pattern of $\widetilde{\mathbf{X}}_c$ consistent with that of $\mathbf{X}_m$, we calculate the ratio of each *MV arrangement* occurred in $\mathbf{X}_m$ based on its missing indicator matrix $\mathbf{S}_m$. In the missing indicator matrix $\mathbf{S}_m$, we define a vector $\mathbf{s}_{m,i} \in \mathbf{S}_m$ as a possible *MV arrangement* to indicate the occurrence of MVs in corresponding observation $o_i \in \mathbf{X}_m$.

Example 8 For the incomplete dataset $\mathbf{X}_m$ in Example 1, there are three MV arrangements, i.e., [1 0 0 0 0 0 0 0], [1 0 0 1 0 0 1 0], [1 0 0 1 0 0 0 0], and the ratios are 3/5, 1/5 and 1/5, respectively.

Next, for each MV arrangement, we randomly select a set of observations based on its ratio from $\mathbf{X}_c$ and replace the values on the incomplete attributes by default values to generate the corrupted input $\widetilde{\mathbf{X}}_c$. With the corrupted input, a MIDIA model is trained for imputing the MVs in the incomplete dataset $\mathbf{X}_m$. In the MV imputation phase, we first initialize the MVs in $\mathbf{X}_m$ with default values generated based on a user-specified scheme, and take the initialized $\mathbf{X}_m$ as the corrupted input. Via the trained MIDIA model, the MVs in $\mathbf{X}_m$ are reconstructed through the encoding and decoding steps.

Example 9 The model training and MV imputation processes based on MIDIA-Batch is shown in Fig. 11. Compared with MIDIA-Sequential, MIIDA-whole only learn a single MIDIA model for the whole MVs.

As introduced earlier, MIDIA-Sequential focuses on imputing the MVs on a single incomplete attribute at a time by splitting the incomplete dataset into several incomplete subsets. It performs well in MV imputation for datasets where the MVs concentrate on one or a few attributes, i.e., there are sufficient complete attributes for use in MV imputation. However, with the general missing pattern, the MVs may occur on any attribute. Consequently, the number of complete attributes in the data is likely to be few. As a result, for an incomplete attribute to be imputed early, the imputation

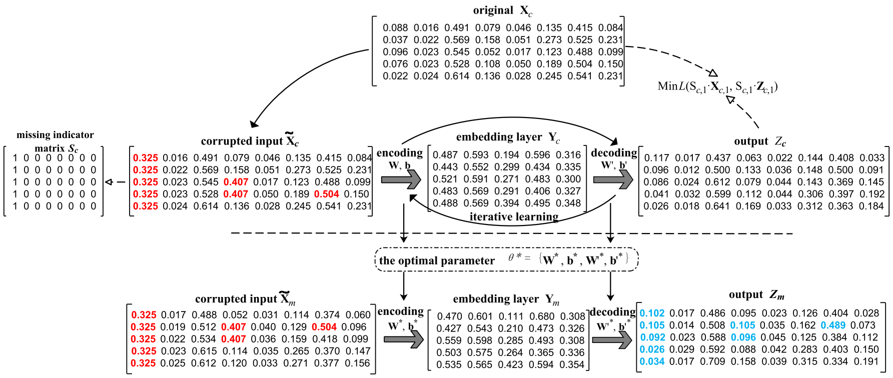


Fig. 11 The model training and MV imputation based on MIDIA-Batch

deviation may be large due to insufficient complete attributes. On the other hand, in MIDIA-Sequential approach, the earlier imputed MVs are used for later imputation, thereby the inaccuracies of the earlier imputed values may be accumulated and amplified in later imputation. Therefore, the imputation results are unsatisfactory based on MIDIA-Sequential for incomplete dataset with general missing pattern. In contrast, MIDIA-Batch imputes the MVs on all incomplete attributes in one batch by training one uniform MIDIA model. Intuitively it is difficult to incorporate all non-linear correlations between MVs and non-MVs in a uniform model, so it is reasonalbe to have the MVs on various incomplete attributes imputed independently if there are sufficient complete attributes to be used. However, with the general missing pattern, the performance of MIDIA-Sequential deteriorates quickly since the number of complete attribute is few. Under this circumstance, MIDIA-Batch alleviate the deterioration by exploring the non-MVs on the incomplete attributes which can be explored for MV imputation. Moerover, under the scenario of univariate missing pattern, since there is only one incomplete attribute in the dataset, the model training and MV imputation processes of MIDIA-Batch is the same with that of MIDIA-Sequential. In summary, as discussed above, both MIDIA-Sequential and MIDIA-Batch perform well for data with univariate missing pattern. MIDIA-Sequential is more capable of handling data with monotone missing pattern, while MIDIA-Batch is more capable of handling data with general missing pattern (which can be demonstrated in experimental study in Sect. 4).

4 Experiments

In this section, we report the experimental evaluation on effectiveness of the proposed approaches. All programs are implemented in Python and the experiments are performed on a PC with 3.4GHz CPU and 16GB RAM.

4.1 Experimental settings

Datasets We employ three real-world datasets: Air Quality, Adult, and Car, as detailed below.

- Air Quality: contains the response of a gas multi-sensor device deployed in a significant polluted area of Italy as introduced in Example 1. It contains 8991 observations where each observation has eight attributes.
- Adult[2]: is a dataset of census information from UCI Machine Learning Repository. The dataset contains about 32,000 observations with 14 attributes including *age, workclass, fnlwgt, education, education-num, matital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-peer-week and native-country.*
- Car[3]: contains 1782 observations where each observation has six attributes, i.e., Buying, Maint, Doors, Persons, Lug_Boot and Safety.

All of the attributes are numerical in the Air Quality, while all of the attributes are categorical in the Car dataset. For the Adult dataset, each observation has 6 numerical attributes and 6 categorical attributes, respectively. In the Air Quality dataset, the MVs naturally exist and the corresponding ground truths are known (provided by a co-located reference certified analyzer), thus it is one of the most commonly used real-world dataset in the study of MV imputation. In the Adult dataset, even though MVs also naturally exist, we cannot evaluate the effectiveness of the proposed algorithms by directly using the real MVs as the ground truths of MVs are unknown. Instead, we remove the observations with inherent MVs from the dataset and consider the remaining data as a complete dataset. In the Car datasets, there is no MVs. To thoroughly evaluate the performance of the proposed MV imputation methods on the Adult and Car datasets, we take the missing ratio as a variant and present the imputation accuracy of various MV imputation methods by varying missing ratio in the following evaluation. Therefore, we consider all the above three datasets as originally clean, and generate different scales of incomplete datasets based on various missing ratios. For example, if missing ratio is 5%, we select 5% observations from the entire dataset randomly to constitute the incomplete dataset to report the imputation results, while the remaining observations constitute the complete dataset to train the learning models.

Additionally, to generate the incomplete dataset $\mathbf{X}_m$ with various missing patterns, we introduce the MVs in different ways detailed below.

- *Univariate missing pattern*: We choose one attribute a_i $(1 \leq i \leq d)$ randomly as the incomplete attribute and mark off all values on the attribute a_i of observations in $\mathbf{X}_m$.
- *Monotone missing pattern*: First, we randomly choose half number of attributes as the incomplete attributes. Next, in $\mathbf{X}_m$, the proportion of MVs on each incomplete attribute progressively decreases from 100% with step length 10%. Moreover, the MVs on incomplete attribute a_i is selected randomly from the observations have been selected by the previous incomplete attribute a_{i-1}. Finally, the values selected

[2] http://archive.ics.uci.edu/ml/datasets/Adult.

[3] http://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

as MVs are marked off. For example, suppose the incomplete attributes randomly chosen from Air Quality dataset (which has eight attributes) are CO, NMHC, NO_x and O_3. Then there are 100%, 90%, 80% and 70% MVs on the incomplete attributes introduced above, respectively. Additionally, the 70% MVs on the attribute O_3 are selected from the observations where the values on incomplete attribute NO_x are missing.

- *General missing pattern*: For each observation in $\mathbf{X}_m$, we randomly choose half number of attributes as the incomplete attributes, and mark off the values on the incomplete attributes.

Measurements To evaluate the performance of various MV imputation methods, we adopt the following measurements for numerical and categorical data types:

- RMSE: We adopt the Root Mean Square Error (RMSE) to measure the imputation deviation between the imputation results and the ground truths for numerical data. The lower RMSE is, the imputation results are closer to the ground truths and thus the imputation has better performance.
- Macro-F: For categorical data, we adopt the *Micro-F* (Sokolova and Lapalme 2009) which is mostly adopted in multi-labeled classification as the performance measure. Specifically, suppose there are c possible values for categorical data, then we have $\textit{macro-F} = \frac{1}{c}\sum_{i=1}^{c} F_i$ where F_i are the *F-measure* of the i-th value. A higher Macro-F indicates the imputation performs better.

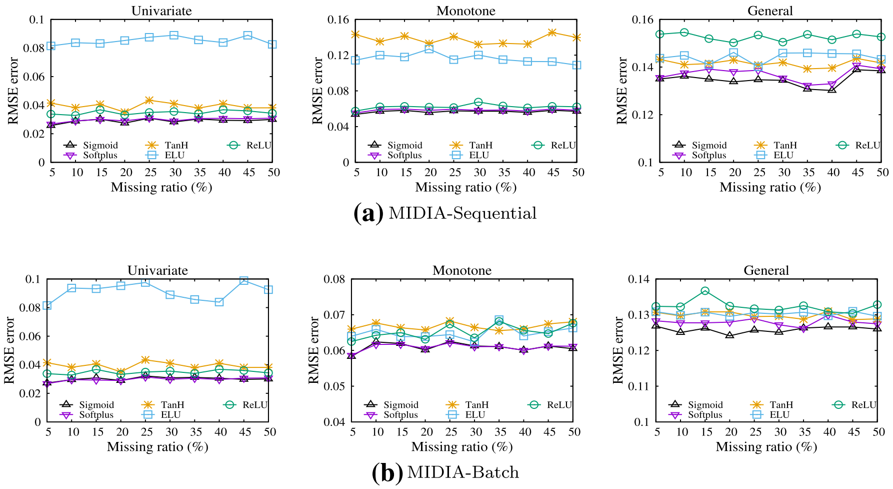
Algorithms for comparison We compare the proposed approaches with the following baseline methods which cover a variety of ways to impute MVs.

- *Mean/Voting* simply imputes the MVs on each incomplete attribute with the mean value/most frequently occurring value of the corresponding attribute for numerical and categorical data, respectively.
- *KNN* (Zhang 2008) uses a distance-weighted average over k neighbors (which similar to the incomplete observation to be imputed) to estimate the MVs in an incomplete observation.
- *Kernel* (Zhu et al. 2011) is similar to the KNN and incorporates various Kernel functions to formalize the dependencies between the incomplete observation and its neighbors.
- *GBKII* (Zhang et al. 2007) imputes MVs through an EM-like iteration imputation method. It differs from KNN imputation in utilizing grey relational grade to measure the neighborhood of MVs.
- *Hot-deck* (Joenssen and Bankhofer 2012) partitions observations into disjoint groups, and predicts MVs using values from one or more similar complete observations (donors) within the same group.
- *Multivariate Linear regression* (MLR) (Raghunathan et al. 2001) argues that there are linear correlations amongst attributes in an observation and imputes the MVs by learning a linear regression model.
- *SVM* (Zhang and Liu 2009; Bertsimas et al. 2017) is a tool of nonlinear regression and classification. We build an SVM model for each incomplete attribute and impute the MVs on each incomplete attribute sequentially with the same order of the proposed MIDIA-Single.

- *Decision Tree (DT)* (Rahman and Islam 2011) is another tool of regression and classification which is similar to the SVM and we implement both SVM and DT by using the library of *sklearn* in python.
- *Low-rank Matrix Recovery* (LMR) (Jing et al. 2016) supposes that a data matrix X can be factorized into two matrices U and V, i.e., $X = UV^T$. By formulating the problem as a matrix rank minimization problem, the optimal U^* and V^* can be estimated by the nuclear-norm minimization. Then the MVs can be estimated based on the imputed matrix which is computed by $X^* = U^*V^{*T}$.
- *Bayesian PCA* (Audigier et al. 2016) formulates the PCA (Principle Component Analysis) as a Bayesian model, instead of using the classical method of finding the covariance matrix of the data. It imputes the MVs by extracting the linear correlations between the MVs and non-MVs.
- *dAE* imputes the MVs in a given dataset by learning a traditional dAE model as introduced in Sect. 2.3.
- *MIDA* (Lovedeep and Wang 2017) imputes the MVs by employing overcomplete representation of dAEs, i.e., there are more units in successive hidden layers compared to the input layer.

Model settings For ease of evaluation and to facilitate faster convergence, we normalize the numerical data based on min-max normalization (Jain and Bhandare 2011) and adopt the one-hot coding to represent the categorical data. Moreover, we repeat 10 times for each test and report the average results to obtain reliable experimental results. For statistical imputation approaches, i.e., kNN, Kernel and GBKII, the parameter k is set as 10 since a low imputation error within an acceptable time range for all three datasets are reached with $k = 10$. For both LMR and Bayesian PCA, the dimensionality of sub-matrix or latent space is specified as the half of the input dimensionality. For approaches based on deep learning model (including MIDIA-Sequential, MIDIA-Batch, dAE and MIDA), each model is trained with epochs 1000, learning rate 0.01 and batch size 256. The MVs synthetically generated are replaced by the mean values or voting values of the corresponding attributes for numerical and categorical data, respectively. Moreover, the *Sigmoid* and *ReLU* are adopted as activation functions for numerical and categorical data respectively, since we find that the proposed approaches have the best performances with the above two activation functions (as illustrated in Sect. 4.2 in detail). Additionally, the MIDIA model in this paper has one hidden (embedding) layer since the imputation accuracy based on such simple model is lower than existing MV imputation approaches. Moreover, for the regular datasets (with low dimensionality), the imputation accuracy of the deep MIDIA model with multiple hidden layers is almost the same with MIDIA model with one hidden layer (as illustrated in Sect. 4.4). As introduced in Lovedeep and Wang (2017), MIDA model has three hidden layers in Encoder,[4] and the i-th hidden layer has 7 units more than the $(i - 1)$-th layer. We adopt the same network structure when we implement the MV imputation based on MIDA model.

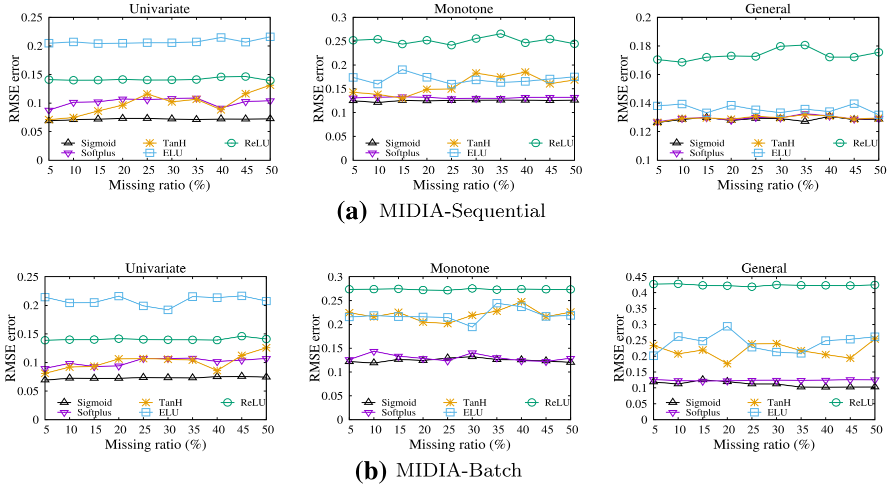
[4] We only consider the number of hidden layers in Encoder since the Decoder is symmetric with the Encoder.



(a) MIDIA-Sequential

(b) MIDIA-Batch

Fig. 12 Imputation accuracy with various activation functions (Air Quality)



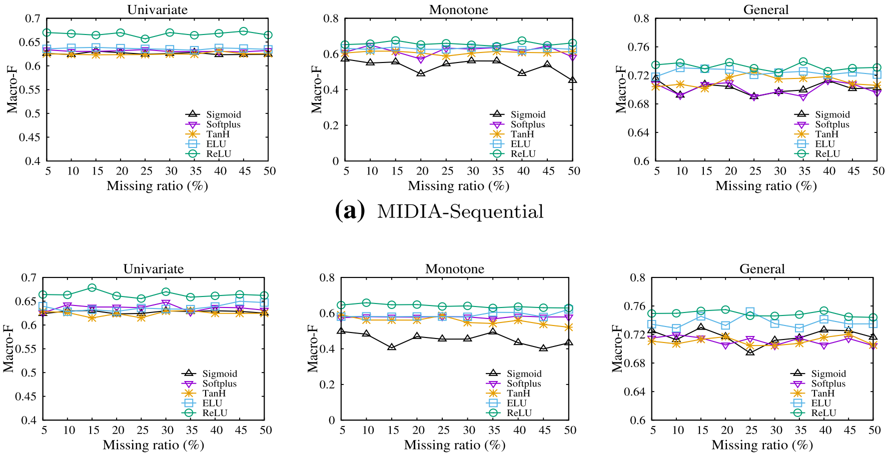
(a) MIDIA-Sequential

(b) MIDIA-Batch

Fig. 13 Imputation accuracy with various activation functions (Adult-Numerical)

4.2 Selection of activation functions

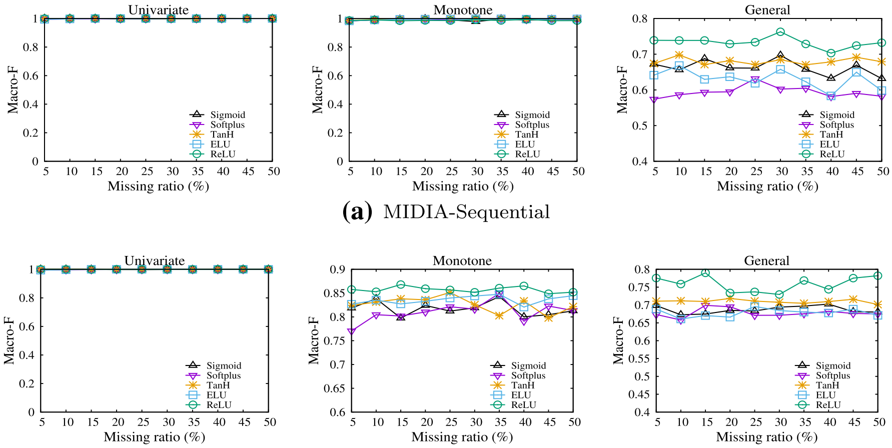
Since the activation function impacts the performances for different datasets and applications, we verify the performances of MIDIA-Sequential and MIDIA-Batch with various activation functions, including Simgoid, TanH, ReLU, Softplus and ELU in this section. Note that Figs. 13 and 14 are the imputation accuracies of numerical attributes and categorical attributes over Adult dataset, respectively. Figures 12, 13, 14 and 15 report the imputation performance of MIDIA-Sequential and MIDIA-Batch with various activation functions by varying missing ratio, respectively. As shown in Figs. 12 and 13, for numerical dataset, both MIDIA-Sequential and MIDIA-Batch



(a) MIDIA-Sequential

(b) MIDIA-Batch

Fig. 14 Imputation accuracy with various activation functions (Adult-Categorical)



(a) MIDIA-Sequential

(b) MIDIA-Batch

Fig. 15 Imputation accuracy with various activation functions (Car)

achive the lowest RMSEs by adopting activation function *Sigmoid* under various missing patterns. In the same way, for categorical dataset, it is obvious that both MIDIA-Sequential and MIDIA-Batch perform best by adopting activation function *ReLU* as shown in Figs. 14 and 15. Therefore, we adopt the *Sigmoid* and *ReLU* as the default activation function for numerical and categorical data, respectively.

4.3 Comparison with existing methods

In this section, we compare our proposed methods with existing MV imputation methods introduced in Sect. 4.1.

Table 2 The RMSE of the imputation results (Air Quality)

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
Univariate										
Mean	0.1466	0.1544	0.1474	0.1455	0.1470	0.1448	0.1451	0.1475	0.1459	0.1447
KNN	0.0292	0.0318	0.0324	0.0326	0.0328	0.0357	0.0351	0.0380	0.0372	0.0394
Kernel	0.0287	0.0318	0.0310	0.0319	0.0344	0.0322	0.0334	0.0333	0.0342	0.0351
GBKII	0.0288	0.0329	0.0313	0.0309	0.0320	0.0320	0.0333	0.0338	0.0331	0.0344
Hot-deck	0.0352	0.0351	0.0337	0.0336	0.0347	0.0351	0.0356	0.0362	0.0359	0.0373
MLR	0.0336	0.0373	0.0380	0.0355	0.0382	0.0358	0.0365	0.0362	0.0360	0.0365
SVM	0.0379	0.0422	0.0399	0.0389	0.0412	0.0409	0.0415	0.0397	0.0405	0.0401
DT	0.0333	0.0337	0.0359	0.0366	0.0382	0.0339	0.0368	0.0367	0.0356	0.0387
LMR	0.1194	0.1182	0.1162	0.1203	0.1164	0.1243	0.1163	0.1173	0.1079	0.1223
B-PCA	0.1018	0.1081	0.1043	0.1007	0.1006	0.1010	0.1028	0.1037	0.1017	0.1028
dAE	0.0863	0.0878	0.0831	0.0894	0.0953	0.0836	0.0970	0.0932	0.0823	0.0970
MIDA	0.1095	0.1153	0.1131	0.1110	0.1195	0.1217	0.1098	0.1195	0.1275	0.1195
MIDIA-S	**0.0257**	**0.0288**	**0.0302**	**0.0274**	**0.0309**	**0.0282**	**0.0311**	**0.0294**	**0.0292**	**0.0300**
MIDIA-B	**0.0272**	**0.0295**	**0.0308**	**0.0289**	**0.0321**	**0.0308**	**0.0302**	**0.0305**	**0.0297**	**0.0299**
Monotone										
Mean	0.1472	0.1545	0.1466	0.1470	0.1498	0.1459	0.1465	0.1478	0.1480	0.1453
KNN	0.0604	0.0652	0.0660	0.0633	0.0648	0.0653	0.0647	0.0638	0.0650	0.0644
Kernel	0.0604	0.0652	0.0660	0.0632	0.0648	0.0652	0.0647	0.0638	0.0649	0.0643
GBKII	0.0970	0.1028	0.0954	0.0976	0.0999	0.0978	0.0980	0.0963	0.0970	0.0984
Hot-deck	0.0814	0.0928	0.0846	0.0845	0.0875	0.0850	0.0849	0.0880	0.0836	0.0880

Table 2 continued

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
MLR	0.1134	0.1201	0.1137	0.1125	0.1161	0.1126	0.1134	0.1135	0.1131	0.1124
SVM	0.0584	0.0640	0.0630	0.0607	0.0631	0.0622	0.0623	0.0620	0.0620	0.0617
DT	0.0770	0.0792	0.0818	0.0797	0.0790	0.0793	0.0800	0.0815	0.0800	0.0826
LMR	0.1379	0.1230	0.1487	0.1348	0.1392	0.1416	0.1270	0.1300	0.1217	0.1200
B-PCA	0.0984	0.1028	0.1000	0.0990	0.1008	0.0982	0.0975	0.1009	0.0990	0.0995
dAE	0.1201	0.1277	0.1157	0.1172	0.1347	0.1195	0.1139	0.1158	0.1249	0.1155
MIDA	0.1319	0.1219	0.1239	0.1272	0.1520	0.1521	0.1162	0.1207	0.1468	0.1255
MIDIA-S	**0.0536**	**0.0570**	**0.0581**	**0.0557**	**0.0575**	**0.0572**	**0.0571**	**0.0558**	**0.0582**	**0.0571**
MIDIA-B	0.0583	0.0623	0.0619	0.0601	0.0625	0.0612	0.0610	0.0600	0.0612	0.0605
General										
Mean	0.1806	0.1818	0.1773	0.1777	0.1805	0.1763	0.1773	0.1789	0.1786	0.1769
KNN	0.0920	0.0936	0.0954	0.0910	0.0917	0.0895	0.0931	0.0947	0.0938	0.0958
Kernel	0.0920	0.0936	0.0954	0.0909	0.0917	0.0895	0.0931	0.0946	0.0937	0.0958
GBKII	0.1362	0.1339	0.1335	0.1334	0.1363	0.1314	0.1332	0.1349	0.1344	0.1346
Hot-deck	0.1301	0.1322	0.1280	0.1320	0.1350	0.1311	0.1276	0.1344	0.1330	0.1376
MLR	0.1456	0.1412	0.1391	0.1398	0.1424	0.1373	0.1403	0.1407	0.1413	0.1392
SVM	0.1491	0.1480	0.1482	0.1470	0.1498	0.1460	0.1478	0.1482	0.1475	0.1471
DT	0.1551	0.1559	0.1574	0.1574	0.1616	0.1598	0.1610	0.1644	0.1615	0.1643
LMR	0.1681	0.1576	0.1522	0.1529	0.1511	0.1551	0.1689	0.1736	0.1550	0.1552
B-PCA	0.1033	0.1045	0.1016	0.1010	0.1009	0.1004	0.1002	0.1033	0.1017	0.1003
dAE	0.1364	0.1373	0.1437	0.1391	0.1375	0.1389	0.1387	0.1401	0.1450	0.1432
MIDA	0.1297	0.1270	0.1262	0.1240	0.1296	0.1250	0.1282	0.1265	0.1275	0.1259
MIDIA-S	0.1038	0.1061	0.1060	0.1023	0.0981	0.1042	0.1067	0.1089	0.1044	0.1038
MIDIA-B	**0.0854**	**0.0857**	**0.0877**	**0.0835**	**0.0859**	**0.0849**	**0.0855**	**0.0869**	**0.0869**	**0.0891**

Based on the experimental results (bold values) have the best performance

Table 3 The RMSE of the imputation results (Adult-Numerical)

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
Univariate										
Mean	0.0799	0.0799	0.0781	0.0795	0.0797	0.0786	0.0799	0.0790	0.0774	0.0791
KNN	0.0765	0.0784	0.0781	0.0804	0.0786	0.0783	0.0775	0.0784	0.0776	0.0779
Kernel	0.0729	0.0743	0.0745	0.0767	0.0759	0.0747	0.0740	0.0752	0.0747	0.0745
GBKII	0.0707	0.0723	0.0726	0.0741	0.0735	0.0726	0.0718	0.0727	0.0725	0.0721
Hot-deck	0.0725	0.0742	0.0745	0.0765	0.0756	0.0743	0.0748	0.0750	0.0749	0.0739
MLR	0.0696	0.0710	0.0716	0.0731	0.0725	0.0713	0.0724	0.0715	0.0713	0.0723
SVM	0.0700	0.0718	0.0722	0.0734	0.0734	0.0729	0.0712	0.0725	0.0720	0.0716
DT	0.0961	0.0965	0.0969	0.0972	0.0989	0.0980	0.1000	0.1003	0.0984	0.0985
LMR	0.1042	0.1041	0.1040	0.1041	0.1040	0.1039	0.1042	0.1040	0.1043	0.1040
B-PCA	0.0770	0.0768	0.0744	0.0768	0.0760	0.0786	0.0771	0.0769	0.0770	0.0761
dAE	0.0833	0.0830	0.0830	0.0837	0.0837	0.0836	0.0837	0.0836	0.0837	0.0837
MIDA	0.0763	0.0764	0.0759	0.0762	0.0761	0.0765	0.0751	0.0755	0.0764	0.0760
MIDIA-S	**0.0688**	**0.0707**	**0.0719**	**0.0713**	**0.0729**	**0.0702**	**0.0709**	**0.0691**	**0.0719**	**0.0709**
MIDIA-B	**0.0692**	**0.0705**	**0.0722**	**0.0712**	**0.0730**	**0.0690**	**0.0712**	**0.0692**	**0.0715**	**0.0704**
Monotone										
Mean	0.1625	0.1651	0.1642	0.1670	0.1696	0.1644	0.1622	0.1667	0.1655	0.1622
KNN	0.1495	0.1507	0.1540	0.1530	0.1515	0.1535	0.1518	0.1525	0.1521	0.1519
Kernel	0.1452	0.1449	0.1472	0.1469	0.1456	0.1477	0.1457	0.1460	0.1450	0.1457
GBKII	0.1430	0.1412	0.1435	0.1427	0.1423	0.1445	0.1417	0.1435	0.1421	0.1431
Hot-deck	0.1445	0.1423	0.1441	0.1430	0.1435	0.1446	0.1429	0.1443	0.1434	0.1439

Table 3 continued

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
MLR	0.1359	0.1346	0.1368	0.1358	0.1363	0.1383	0.1360	0.1370	0.1361	0.1365
SVM	0.1239	0.1221	0.1242	0.1236	0.1251	0.1258	0.1233	0.1243	0.1225	0.1248
DT	0.1685	0.1753	0.1704	0.1673	0.1680	0.1750	0.1701	0.1715	0.1672	0.1684
LMR	0.2554	0.2492	0.2450	0.2430	0.2528	0.2518	0.2370	0.2520	0.2506	0.2600
B-PCA	0.1438	0.1456	0.1422	0.1448	0.1417	0.1481	0.1471	0.1480	0.1458	0.1419
dAE	0.2573	0.2648	0.2678	0.2443	0.2631	0.2497	0.2692	0.2582	0.2548	0.2625
MIDA	0.1599	0.1561	0.1609	0.1508	0.1521	0.1543	0.1544	0.1562	0.1567	0.1568
MIDIA-S	**0.1219**	**0.1189**	**0.1254**	**0.1246**	**0.1251**	**0.1265**	**0.1262**	**0.1254**	**0.1232**	**0.1202**
MIDIA-B	0.1248	0.1212	0.1265	0.1243	0.1291	0.1322	0.1262	0.1260	0.1251	0.1261
General										
Mean	0.1774	0.1830	0.1850	0.1854	0.1821	0.1933	0.1905	0.1864	0.1847	0.1869
KNN	0.1676	0.1730	0.1697	0.1700	0.1705	0.1708	0.1684	0.1702	0.1688	0.1690
Kernel	0.1615	0.1650	0.1622	0.1616	0.1622	0.1629	0.1607	0.1616	0.1601	0.1609
GBKII	0.1515	0.1525	0.1534	0.1520	0.1530	0.1537	0.1516	0.1501	0.1518	0.1490
Hot-deck	0.1519	0.1530	0.1534	0.1521	0.1532	0.1539	0.1520	0.1507	0.1522	0.1500
MLR	0.1409	0.1440	0.1445	0.1432	0.1438	0.1437	0.1436	0.1435	0.1420	0.1428
SVM	0.1327	0.1354	0.1353	0.1342	0.1350	0.1346	0.1347	0.1345	0.1335	0.1348
DT	0.1634	0.1650	0.1608	0.1596	0.1614	0.1633	0.1614	0.1626	0.1586	0.1620
LMR	0.1942	0.1775	0.1876	0.1852	0.1955	0.2071	0.1998	0.1941	0.2045	0.1932
B-PCA	0.1476	0.1477	0.1475	0.1592	0.1488	0.1418	0.1427	0.1483	0.1434	0.1470
dAE	0.1812	0.1813	0.1776	0.1736	0.1858	0.1868	0.1727	0.1898	0.1911	0.1828
MIDA	0.1581	0.1600	0.1585	0.1608	0.1600	0.1581	0.1585	0.1599	0.1587	0.1587
MIDIA-S	0.1263	0.1285	0.1298	0.1278	0.1293	0.1291	0.1271	0.1306	0.1283	0.1287
MIDIA-B	**0.1185**	**0.1124**	**0.1254**	**0.1196**	**0.1122**	**0.1123**	**0.1025**	**0.1021**	**0.1024**	**0.1026**

Based on the experimental results (bold values) have the best performance

Table 4 The Macro-F of the imputation results (Adult-Categorical)

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
Univariate										
Voting	0.5357	0.5303	0.5303	0.5277	0.5342	0.5259	0.5281	0.5299	0.5301	0.5241
KNN	0.6431	0.6379	0.6417	0.6331	0.6421	0.6332	0.6340	0.6409	0.6422	0.6336
Kernel	0.6247	0.6285	0.6301	0.6230	0.6311	0.6255	0.6222	0.6269	0.6219	0.6152
GBKII	0.6219	0.6074	0.6143	0.6122	0.6298	0.6178	0.6245	0.6313	0.6246	0.6237
Hot-deck	0.5829	0.5688	0.5799	0.5708	0.5823	0.5740	0.5669	0.5885	0.5837	0.5809
MLR	0.6353	0.6303	0.6304	0.6278	0.6346	0.6256	0.6278	0.6297	0.6302	0.6240
SVM	0.5218	0.5052	0.4951	0.5036	0.5154	0.5140	0.5108	0.5201	0.5403	0.5155
DT	0.6045	0.6003	0.6140	0.6186	0.6242	0.6065	0.6146	0.6054	0.6132	0.6088
LMR	0.3218	0.3120	0.3118	0.3100	0.3136	0.3210	0.3163	0.3121	0.3100	0.3100
B-PCA	0.6357	0.6303	0.6303	0.6277	0.6342	0.6259	0.6281	0.6300	0.6301	0.6241
dAE	0.6357	0.6399	0.6303	0.6435	0.6349	0.6336	0.6280	0.6517	0.6325	0.6241
MIDA	0.6173	0.6260	0.6290	0.6276	0.6310	0.6256	0.6395	0.6295	0.6300	0.6229
MIDIA-S	**0.6699**	**0.6675**	**0.6646**	**0.6698**	**0.6568**	**0.6700**	**0.6643**	**0.6684**	**0.6729**	**0.6649**
MIDIA-B	**0.6640**	**0.6633**	**0.6785**	**0.6612**	**0.6556**	**0.6699**	**0.6586**	**0.6613**	**0.6642**	**0.6622**
Monotone										
Voting	0.3814	0.3875	0.3900	0.3888	0.3866	0.3846	0.3868	0.3861	0.3879	0.3837
KNN	0.6719	0.6781	0.6763	0.6740	0.6786	0.6749	0.6755	0.6756	0.6748	0.6726
Kernel	0.6595	0.6651	0.6584	0.6596	0.6568	0.6566	0.6528	0.6566	0.6576	0.6560
GBKII	0.5919	0.5909	0.5834	0.5989	0.5898	0.6043	0.6266	0.6223	0.6026	0.6198
Hot-deck	0.5549	0.5751	0.5585	0.5632	0.5573	0.5609	0.5847	0.5787	0.5848	0.5772
MLR	0.3580	0.3557	0.3567	0.3567	0.3531	0.3567	0.3551	0.3541	0.3557	0.3541
SVM	0.3377	0.3330	0.3384	0.3310	0.3328	0.3367	0.3407	0.3424	0.3473	0.3336
DT	0.6368	0.6511	0.6377	0.6359	0.6369	0.6357	0.6329	0.6383	0.6365	0.6307

Table 4 continued

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
LMR	0.3490	0.3540	0.3598	0.3473	0.3461	0.3540	0.3416	0.3475	0.3480	0.3473
B-PCA	0.3805	0.3915	0.3914	0.3884	0.3894	0.3851	0.3871	0.3861	0.3879	0.3838
dAE	0.5675	0.5551	0.5584	0.5512	0.5511	0.5619	0.5541	0.5583	0.5636	0.5630
MIDA	0.5564	0.5600	0.5269	0.5264	0.5264	0.5256	0.5296	0.5252	0.5306	0.5090
MIDIA-S	**0.6524**	**0.6575**	**0.6759**	**0.6528**	**0.6596**	**0.6517**	**0.6431**	**0.6749**	**0.6482**	**0.6617**
MIDIA-B	0.6450	0.6581	0.6466	0.6478	0.6373	0.6405	0.6290	0.6361	0.6303	0.6289
General										
Voting	0.5315	0.5418	0.5459	0.5461	0.5413	0.5490	0.5437	0.5450	0.5492	0.5420
KNN	0.5810	0.5833	0.5911	0.5941	0.5903	0.5876	0.5937	0.5922	0.5922	0.5903
Kernel	0.5800	0.5792	0.5909	0.5913	0.5866	0.5852	0.5920	0.5893	0.5888	0.5863
GBKII	0.5574	0.5946	0.5902	0.5973	0.5714	0.6015	0.5985	0.6099	0.5927	0.6097
Hot-deck	0.5256	0.5638	0.5588	0.5605	0.5418	0.5698	0.5586	0.5704	0.5643	0.5599
MLR	0.4892	0.4972	0.4943	0.4964	0.4914	0.4934	0.4895	0.4922	0.4939	0.4934
SVM	0.4479	0.4668	0.4423	0.4451	0.4573	0.4479	0.4403	0.4582	0.4602	0.4552
DT	0.6980	0.7080	0.6979	0.7073	0.7224	0.7023	0.7021	0.7112	0.7000	0.7062
LMR	0.5654	0.5564	0.5633	0.5582	0.5619	0.5630	0.5648	0.5502	0.5441	0.5441
B-PCA	0.5821	0.5996	0.5945	0.5981	0.5942	0.5900	0.5934	0.5944	0.5925	0.5921
dAE	0.6947	0.6956	0.7084	0.7266	0.7136	0.7162	0.6948	0.6952	0.7098	0.7079
MIDA	0.5885	0.5996	0.5947	0.5980	0.5941	0.5753	0.5861	0.5945	0.5927	0.5921
MIDIA-S	0.7347	0.7376	0.7292	0.7382	0.7300	0.7238	0.7394	0.7258	0.7301	0.7310
MIDIA-B	**0.7495**	**0.7497**	**0.7531**	**0.7549**	**0.7462**	**0.7460**	**0.7479**	**0.7535**	**0.7448**	**0.7440**

Based on the experimental results (bold values) have the best performance

Table 5 The Macro-F of the imputation results (Car)

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
Univariate										
Voting	0.2097	0.2238	0.2366	0.2557	0.2204	0.2277	0.1912	0.1977	0.2048	0.2795
KNN	0.4965	0.4753	0.4314	0.4405	0.4382	0.4702	0.4330	0.4452	0.4484	0.4844
Kernel	0.4282	0.4619	0.3877	0.3908	0.3971	0.4101	0.3877	0.3932	0.4089	0.3978
GBKII	0.4831	0.4934	0.4353	0.4560	0.4818	0.4881	0.4618	0.4380	0.4950	0.4677
Hot-deck	0.3454	0.3318	0.3454	0.3572	0.3511	0.3050	0.3330	0.3661	0.3062	0.3621
MLR	0.3171	0.2658	0.2991	0.2853	0.2815	0.2830	0.2862	0.2826	0.2963	0.3408
SVM	0.3319	0.3155	0.3167	0.3226	0.3167	0.3385	0.3585	0.3420	0.3499	0.3265
DT	0.2454	0.2558	0.2785	0.2453	0.2855	0.2649	0.2416	0.2500	0.3018	0.3003
LMR	0.1948	0.1822	0.1628	0.2353	0.2281	0.3981	0.1639	0.3393	0.1797	0.2282
B-PCA	0.5884	0.5337	0.5211	0.5571	0.5455	0.5428	0.5499	0.5446	0.5294	0.5796
dAE	0.9940	0.9941	1.0000	1.0000	1.0000	0.9990	0.9991	1.0000	1.0000	1.0000
MIDA	0.9883	1.0000	1.0000	0.9980	0.9983	1.0000	1.0000	1.0000	1.0000	1.0000
MIDIA-S	**1.0000**	**1.0000**	**1.0000**	**0.9985**	**0.9988**	**0.9990**	**0.9991**	**1.0000**	**0.9993**	**1.0000**
MIDIA-B	**1.0000**	**1.0000**	**0.9980**	**1.0000**	**0.9988**	**1.0000**	**0.9991**	**1.0000**	**1.0000**	**1.0000**
Monotone										
Voting	0.1778	0.1558	0.1465	0.1428	0.1469	0.1476	0.1599	0.1584	0.1523	0.1519
KNN	0.2970	0.2825	0.2655	0.3125	0.3132	0.3020	0.2950	0.3039	0.3056	0.2976
Kernel	0.2898	0.2863	0.2721	0.3170	0.3123	0.2863	0.3016	0.2995	0.3061	0.2974
GBKII	0.3235	0.3293	0.2961	0.3281	0.3289	0.3001	0.3093	0.3065	0.3449	0.3005
Hot-deck	0.2157	0.2103	0.2363	0.2755	0.3025	0.2860	0.2885	0.3016	0.2939	0.2721

Table 5 continued

	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
MLR	0.2075	0.2038	0.2109	0.2262	0.2610	0.2602	0.2257	0.2668	0.2549	0.2555
SVM	0.2738	0.2406	0.2279	0.2330	0.2596	0.2226	0.2589	0.2380	0.2621	0.2739
DT	0.1979	0.2508	0.2511	0.2917	0.2954	0.3044	0.3003	0.3124	0.3166	0.3302
LMR	0.1379	0.1423	0.1504	0.1534	0.1739	0.1794	0.1727	0.2300	0.1521	0.1705
B-PCA	0.5466	0.4422	0.4441	0.4098	0.4411	0.5083	0.4218	0.4425	0.4286	0.4787
dAE	0.6712	0.6745	0.5985	0.7110	0.6740	0.6721	0.5913	0.7116	0.6503	0.7186
MIDA	0.7485	0.7822	0.7860	0.7806	0.8169	0.7633	0.7938	0.8002	0.7909	0.8076
MIDIA-S	**0.9825**	**0.9935**	**0.9928**	**0.9957**	**0.9948**	**0.9942**	**0.9950**	**0.9983**	**0.9961**	**0.9978**
MIDIA-B	0.8577	0.8528	0.8680	0.8592	0.8567	0.8518	0.8604	0.8651	0.8487	0.8519
General										
Voting	0.1700	0.1545	0.1901	0.1519	0.1581	0.1572	0.1595	0.1666	0.1704	0.1535
KNN	0.2535	0.2400	0.2795	0.2820	0.2692	0.2585	0.2601	0.2811	0.2809	0.2693
Kernel	0.2678	0.2655	0.2817	0.2844	0.2571	0.2436	0.2594	0.2758	0.2706	0.2834
GBKII	0.3200	0.2782	0.2807	0.3014	0.2819	0.2867	0.2969	0.2805	0.3098	0.2856
Hot-deck	0.2817	0.2571	0.2592	0.2844	0.2624	0.2708	0.2687	0.2733	0.2996	0.2866
MLR	0.2086	0.2014	0.2097	0.1874	0.1959	0.2015	0.2159	0.2039	0.2139	0.1819
SVM	0.2117	0.1984	0.2093	0.2517	0.2440	0.2359	0.2323	0.2289	0.2179	0.2249
DT	0.2728	0.2760	0.2578	0.2916	0.2639	0.2988	0.2541	0.2521	0.3238	0.3199
LMR	0.1681	0.1576	0.1522	0.1529	0.1511	0.1551	0.1689	0.1736	0.1550	0.1552
B-PCA	0.4135	0.4113	0.5019	0.4510	0.4014	0.3982	0.4118	0.4557	0.4427	0.4403
dAE	0.6071	0.5634	0.6580	0.6180	0.6418	0.6568	0.6554	0.6610	0.6708	0.6528
MIDA	0.7293	0.7083	0.6955	0.7032	0.7139	0.7296	0.7202	0.7076	0.7110	0.7080
MIDIA-S	0.7388	0.7385	0.7385	0.7287	0.7336	0.7626	0.7286	0.7030	0.7241	0.7321
MIDIA-B	**0.7754**	**0.7586**	**0.7899**	**0.7338**	**0.7367**	**0.7295**	**0.7686**	**0.7439**	**0.7752**	**0.7821**

Based on the experimental results (bold values) have the best performance

Table 2 summarizes the imputation performance of all compared approaches on Air Quality dataset with various missing patterns As shown, since the Mean imputation fills the MVs on an attribute by the mean of the available observations, where the variability of the data is ignored, the imputation accuracy is the worst (i.e., the RMSE is the highest) in most cases. For the neighbor-based imputation methods (i.e., KNN, Kernel, GBKII and Hot-deck), they impute the MVs by employing the inter-correlations amongst observations, while the proposed MIDIA-based imputation methods impute the MVs by employing the intra-correlations amongst attributes in the same observation. Even though there is no theoretical support that one is absolutely better than the other, the experimental results show that their RMSEs are higher than the proposed MIDIA-based methods. On the other hand, for the MLR, LMR and Bayesian PCA, they impute the MVs by exploring the linear correlations between the MVs and non-MVs, where the complex correlations of the data tend to be underestimated, which results in unsatisfactory imputation accuracies. Although SMV and DT can explore the non-linear correlations between the MVs and non-MVs based on kernel functions, the choice of kernel function is usually guided by experience, thus suffering the same problem with the linear regression models mentioned above. Finally, the proposed MIDIA-based approaches achieve the lowest RMSE, as the proposed MIDIA model effectively captures the non-linear correlations between MVs and non-MVs in the data, which is more powerful on MV imputation compared with existing approaches that explore the linear dependencies between MVs and non-MVs. Moreover, by designing effective generation strategy for corrupted input and minimizing the reconstruction error between MVs and ground truths in the training process, the learned MIDIA model is more effective than dAE and MIDA.

Additionally, under the scenario of univariate missing pattern, the RMSEs of MIDIA-Sequential and MIDIA-Batch are almost the same. The reason is that there is only one single incomplete attribute and all values on the incomplete attribute are missing, which incurs that the model training processes of MIDIA-Sequential and MIDIA-Batch reduce to be the same. Under the scenario of monotone missing pattern, the RMSE of MIDIA-Sequential is lower than that of MIDIA-Batch, because with monotone missing pattern, there are enough complete attributes used for MV imputaiton, and MIDIA-Sequential can avoid the effect of MVs on other incomplete attributes when it focuses on the imputation for an incomplete attribute. However, if there is no enough complete attributes, the imputation results of MIDIA-Sequential are likely to deteriorate, which incurs that the RMSE of MIDIA-Sequential is higher than that of MIDIA-Batch under the scenario of general missing pattern, as shown in Table 2. With the same line, similar results are observed in Tables 3, 4 and 5 where the experimental evaluation conducts on datasets Adult and Car respectively. Moreover, the results obtained based on the three real-world datasets verify that the proposed approaches can support both numerical and categorical data.

On the other hand, based on Tables 2, 3, 4 and 5, we can observe that with the increase of missing ratio, the imputation accuracies of existing neighbor-based imputation methods (including KNN, Kernel, GBKII and Hot-deck) progressively decrease. With a high missing ratio, there are more incomplete observations and less complete observations, thereby the number of available neighbors used for MV imputation becomes small and incurs a low imputation accuracy. Notable that since the data size of Adult dataset is large (32,000 observations), the available neighbors used for MV imputation are sufficient even though the missing ratio is high, thereby the imputation accuracies of existing neighbor-based imputation methods are stable. By comparison, for the remaining imputation methods, the imputation accuracies are stable with the increase of missing ratio, and the RMSEs of the proposed MIDIA-Sequential and MIDIA-Batch are significantly lower than baselines, which indicates that the proposed approaches are effective and capable of handling datasets with small samples.

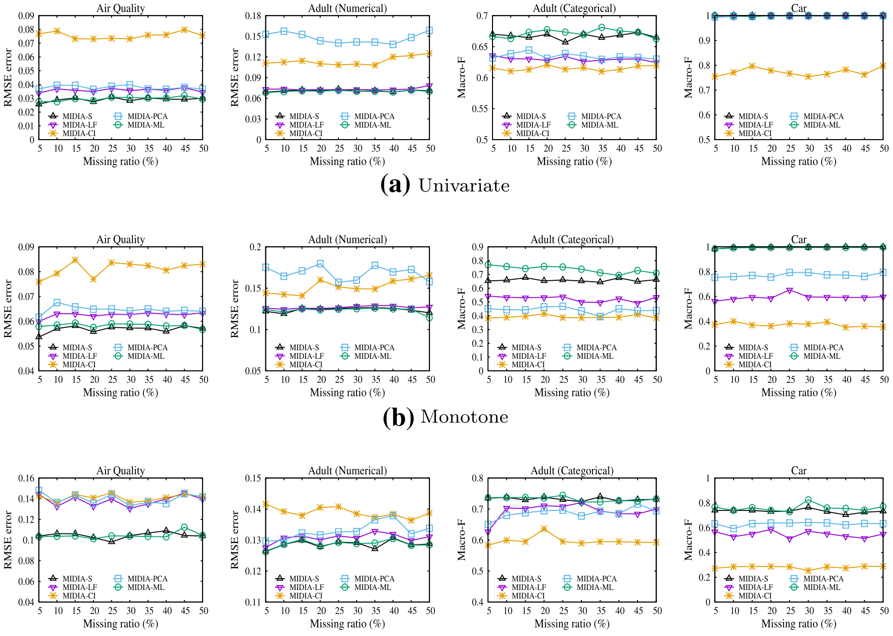
In summary, as illustrated in Tables 2, 3, 4 and 5 over the Air Quality, Adult and Car datasets, the proposed MIDIA-based imputation approaches always achieve the best performance in imputation accuracy. Moreover, both MIDIA-Sequential and MIDIA-Batch perform well for dataset with univariate missing pattern. MIDIA-Sequential performs better than MIDIA-Batch for dataset with monotone missing pattern, while MIDIA-Batch performs better than MIDIA-Sequential for dataset with general missing pattern.

4.4 Ablation study

As illustrated earlier, the main modifications of the proposed MIDIA mainly focus on the corrupted input generation and objective function. To evaluate the effectiveness of each modification in improving the imputation accuracy, we implement the MV imputation approaches based on the MIDIA model by retaining one aspect of modification, including (1) MIDIA-LF, only retaining the modification of objective function; (2) MIDIA-CI, only retaining the modification of corrupted input generation. Moreover, to evaluate the effectiveness of the neural network model, we implement the (3) MIDIA-PCA, replacing the non-linear activation functions in MIDIA with the linear activation function to approximate PCA, and (4) MIDIA-ML, extending the MIDIA model with one hidden layer to three hidden layers.

Figure 16 illustrates the performance of various algorithms based on MIDIA-Sequential with three missing patterns over three real-world datasets. As shown, the imputation accuracies of algorithms with only one aspect of modification are lower than that of the proposed approach with thorough modifications, which verifies that each aspect of modification contributes to the improvement of imputation accuracy. Moreover, with the same model structure, we observe that the imputation accuracy of MIDIA-PCA is much lower than that of the proposed MIDIA-Sequential (MIDIA-S), because MIDIA-S imputes the MVs in the data by exploring the non-linear correlations between MVs and non-MVs, which is more competent for the data with complex and unexplainable structures. Finally, as shown in Fig. 16 (Fig. 16 in response), the imputation accuracies of MIDIA-ML and MIDIA-S are almost the same. The reason is that the dimensionality and sample size of datasets adopted in the experiments are small or moderate.

With low-dimensional data, MIDIA model with one-hidden layer is capable of deeply exploring the correlations of the data, while the deep MIDIA model cannot give the rein to its advantages. We believe that deep MIDIA model with multiple hidden layers is more competent for handling high-dimensional and large-scale sample dataset, which is an interesting future study. In this paper, we mainly focus on the MV

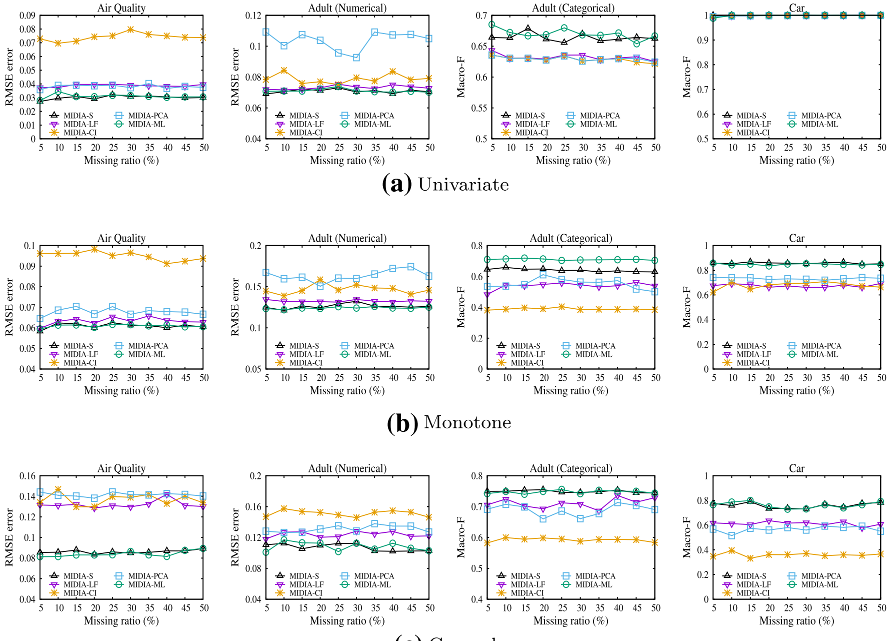


(a) Univariate

(b) Monotone

(c) General

Fig. 16 Performance of each part of modification over MIDIA-Sequentical

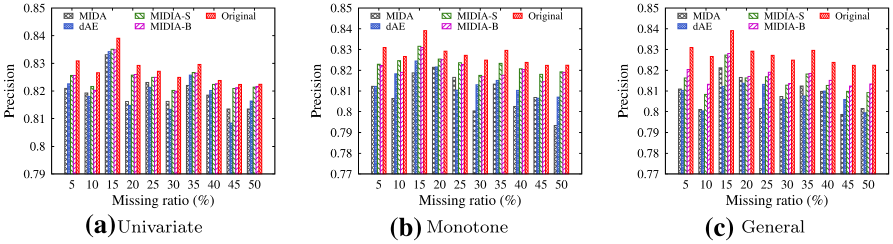


(a) Univariate

(b) Monotone

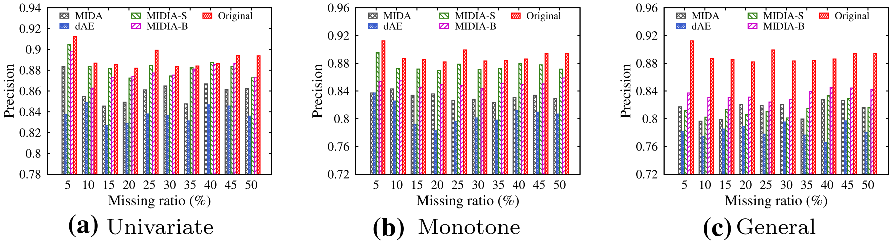
(c) General

Fig. 17 Performance of each part of modification over MIDIA-Batch



(a) Univariate **(b)** Monotone **(c)** General

Fig. 18 The classification accuracy (Adult)



(a) Univariate **(b)** Monotone **(c)** General

Fig. 19 The classification accuracy (Car)

imputation for the regular datasets. In the same line, similar results can be observed in Fig. 17 where the algorithms are implemented based on the MIDIA-Batch.

4.5 Applications in classification

To further validate the effectiveness of our proposed methods, we consider a real application of classification (Liu and Yu 2005) on datasets Adult and Car. Based on the imputed dataset, a softmax classifier is directly implemented. Figures 18 and 19 reports the accuracy of classification over the original dataset (denoted by Original in the figure), the imputed dataset by dAE, MIDA, MIDIA-Sequential and MIDIA-Batch (denoted by MIDIA-S and MIDIA-B respectively in the figure), respectively. Note that we do not show the results of other approaches as the MV imputation accuracies of them are obviously smaller than dAE, MIDA, MIDIA-Sequential and MIDIA-Batch in most cases. The classification accuracy of the original data is the best because there is no MVs in original dataset. It is not surprising that the classification accuracies of MIDIA-Sequential and MIDIA-Batch are higher than dAE and MIDA, largely because they have a higher imputation accuracies. The results further demonstrate the effectiveness of the proposed approaches.

5 Conclusion

In this paper, we propose a new unsupervised learning model, named MIDIA, tailored for MV imputation. By considering various missing data patterns, we propose two MV imputation approaches based on the proposed MIDIA model, namely MIDIA-

Sequential and MIDIA-Batch, where both approaches perform well for univariate missing pattern, and MIDIA-Sequential is more competent for monotone missing pattern while MIDIA-Batch performs better for general missing pattern. Experimental results on real-world datasets show that the proposed approaches significantly improve the imputation accuracy compared with existing methods.

Acknowledgements This work is supported by the China Postdoctoral Science Foundation (2019M661077), the National Science Foundation (Grant No. IIS-1717084), the National Natural Science Foundation of China (Grant Nos. 61772102, 61751205), and the Liaoning Revitalization Talents Program (XLYC1807158).

References

Aittokallio T (2010) Dealing with missing values in large-scale studies: microarray data imputation and beyond. Brief Bioinform 11(2):253–264

Anagnostopoulos C, Triantafillou P (2014) Scaling out big data missing value imputations: pythia vs. godzilla. In: Proceedings of ACM international conference on knowledge discovery and data mining, pp 651–660

Andridge RR, Little RJA (2010) A review of hot deck imputation for survey non-response. Int Stat Rev 78(1):40–64

Audigier V, Husson F, Josse J (2016) Multiple imputation for continuous variables using a bayesian principal component analysis. J Stat Comput Simul 86(11):2140–2156

Baldi P (2012) Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning, pp 37–50

Bergstra J, Desjardins G, Lamblin P, Bengio Y (2009) Quadratic polynomials learn better image features. Technical report, p 1337

Bertsimas D, Pawlowski C, Zhuo YD (2017) From predictive methods to missing data imputation: an optimization approach. J Mach Learn Res 18(1):7133–7171

Borovicka T, Jirina-Jr M, Kordik P, Jirina M (2012) Selecting representative data sets. In: Advances in data mining knowledge discovery and applications, pp 43–70

Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp 177–186

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the em algorithm. J R Stat Soc Ser B (Methodol) 39(1):1–38

Dong X, Gabrilovich E, Heitz G et al (2014) Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of ACM international conference on knowledge discovery and data mining, pp 601–610

Gharibshah Z, Zhu XQ, Hainline A, Conway M (2020) Deep learning for user interest and response prediction in online display advertising. Data Sci Eng 5(1):12–26

Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of international conference on artificial intelligence and statistics, pp 249–256

Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Proceedings of international conference on artificial intelligence and statistics, pp 315–323

Han J, Moraga C (1995) The influence of the sigmoid function parameters on the speed of backpropagation learning. In: Proceedings of international workshop on artificial neural networks, pp 195–201

Jain YK, Bhandare SK (2011) Min max normalization based data perturbation method for privacy protection. Int J Comput Commun Technol 2(8):45–50

Jing XY, Qi FM, Wu F, Xu BW (2016) Missing data imputation based on low-rank recovery and semi-supervised regression for software effort estimation. In: Proceedings of IEEE/ACM international conference on software engineering, pp 607–618

Joenssen DW, Bankhofer U (2012) Hot deck methods for imputing missing data—the effects of limiting donor usage. In: International workshop on machine learning and data mining in pattern recognition, pp 63–75

Jonathan ACS, White IR, Carlin JB, Spratt M, Royston P, Kenward MG, Wood AM, Carpenter JR (2009) Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. BMJ Br Med J 339(7713):157–160

Kim KY, Kim BJ, Yi GS (2004) Reuse of imputed data in microarray analysis increases imputation efficiency. BMC Bioinform 5:160

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

Liu H, Yu L (2005) Toward integrating feature selection algorithms for classification and clustering. IEEE Trans Knowl Discov Eng 17(4):491–502

Lovedeep G, Wang K (2017) Multiple imputation using deep denoising autoencoders. CoRR arXiv:1705.02737

Magnani M (2004) Techniques for dealing with missing data in knowledge discovery tasks. Obtido 15(01):2007. http://magnanim.web.cs.unibo.it/index.html

McNeish D (2017) Missing data methods for arbitrary missingness with small samples. J Appl Stat 44(1):24–39

Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of international conference on international conference on machine learning, pp 807–814

Qin Y, Zhang S, Zhu X et al (2009) POP algorithm: Kernel-based imputation to treat missing values in knowledge discovery from databases. Expert Syst Appl 36(2):2794–2804

Raghunathan TE, Lepkowski JM, Hoewyk JV, Solenberger P (2001) A multivariate technique for multiply imputing missing values using a sequence of regression models. Survey Methodol 27(1):85–96

Rahman G, Islam Z (2011) A decision tree-based missing value imputation technique for data preprocessing. In: Proceedings of Australasian data mining conference, pp 41–50

Sinclair JM, Wilkes GA, Krebs WA (2001) Collins concise dictionary. HarperCollins, New York

Sokolova M, Lapalme G (2009) A systematic analysis of performance measures for classification tasks. Inf Process Manag 45(4):427–437

Troyanskaya OG, Cantor MN, Sherlock G et al (2001) Missing value estimation methods for DNA microarrays. Bioinformatics 17(6):520–525

Verboven S, Branden KV, Goos P (2007) Sequential imputation for missing values. Comput Biol Chem 31(5–6):320–327

Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of international conference on machine learning, pp 1096–1103

Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11(12):3371–3408

Vito SD, Massera E, Piga M et al (2008) On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sens Actuators B Chem 129(2):750–757

Wang QH, Rao JNK (2002a) Empirical likelihood-based inference in linear models with missing data. Scand J Stat 29(3):563–576

Wang QH, Rao JNK (2002b) Empirical likelihood-based inference under imputation for missing response data. Ann Stat 30(3):896–924

Yuan YC (2010) Multiple imputation for missing data: concepts and new development, vol 49. SAS Institute Inc, Rockville, pp 1–11

Zhang S (2008) Parimputation: from imputation and null-imputation to partially imputation. IEEE Intell Inform Bull 9(1):32–38

Zhang Y, Liu YC (2009) Data imputation using least squares support vector machines in urban arterial streets. IEEE Signal Process Lett 16(5):414–417

Zhang CQ, Zhu XF, Zhang JL, Qin YS, Zhang SC (2007) GBKII: an imputation method for missing values. In: Proceedings of Pacific-Asia conference on knowledge discovery and data mining, pp 1080–1087

Zhang X, Song X, Wang H et al (2008) Sequential local least squares imputation estimating missing value of microarray data. Comput Biol Med 38(10):1112–1120

Zhou XB, Wang XD, Dougherty ER (2003) Construction of genomic networks using mutual-information clustering and reversible-jump markov-chain-monte-carlo predictor design. Signal Process 83(4):745–761

Zhu X, Zhang S, Jin Z et al (2011) Missing value estimation for mixed-attribute data sets. IEEE Trans Knowl Data Eng 23(1):110–121

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Qian Ma[1] · **Wang-Chien Lee**[2] · **Tao-Yang Fu**[2] · **Yu Gu**[3] · **Ge Yu**[3]

Wang-Chien Lee
wlee@cse.psu.edu

Tao-Yang Fu
txf225@cse.psu.edu

Yu Gu
guyu@mail.neu.edu.cn

Ge Yu
yuge@mail.neu.edu.cn

[1] College of Information Science and Technology, Dalian Maritime University, Dalian, China

[2] Department of Computer Science and Engineering, The Pennsylvania State University, State College, USA

[3] School of Computer Science and Engineering, Northeastern University, Shenyang, China