Knowledge-Based Systems 182 (2019) 104838

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Imputation of missing data with neural networks for classification ☆

Suvra Jyoti Choudhury, Nikhil R. Pal *

Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700108, India

ARTICLE INFO

Article history:
Received 5 December 2018
Received in revised form 6 July 2019
Accepted 8 July 2019
Available online 17 July 2019

Keywords:
Classification
Data imputation
Gradient decent
Missing attribute value
Neural network

ABSTRACT

We propose a mechanism to use data with missing values for designing classifiers which is different from predicting missing values for classification. Our imputation method uses an auto-encoder neural network. We make an innovative use of the training data without missing values to train the auto-encoder so that it is better equipped to predict missing values. It is a two-stage training scheme. Unlike most of the existing auto-encoder based methods which use a bottleneck layer for missing data handling, we justify and use a latent space of much higher dimension than that of the input. Now to design a classifier using a training set with missing values, we use the trained auto-encoder to predict missing values based on the hypothesis that a good choice for a missing value would be the one which can reconstruct itself via the auto-encoder. For this we make an initial guess of the missing value using the nearest neighbor rule and then refine the missing value minimizing the reconstruction error. We train several classifiers using the union of the imputed instances and the remaining training instances without missing values. We also train another classifier of the same type with the same configuration using the corresponding complete dataset. The performances of these classifiers are compared. We compare the proposed method with eight state-of-the-art imputation techniques using fourteen datasets and eight classification strategies.

1. Introduction

In today's world missing data is an important problem while dealing with real world situations [1]. If a data set has some feature vectors with one or more (but not all) missing values, the data set is called an incomplete data set for missing values. Let, $\mathbf{x}_k \in \mathbf{X} \subseteq R^p$ be the kth data point and $\mathbf{X}$ be the data set. We call $\mathbf{X}$ an incomplete data set, if for one or more k, $\mathbf{x}_k$ has $0 < q < p$ missing values. Note that q may be different for different k. In many real life situations we encounter data with missing values; examples include (1) control (traffic monitoring [2], industrial processes [3], management of telecommunications and computer networks [4] etc.), (2) wireless sensor networks [5,6], (3) speech recognition [7,8], (4) business applications [9,10], and (5) medical diagnosis [11–14]. Thus, effective missing data handling techniques have many applications. For example, in [14] authors used missing data imputation for Breast Cancer Prognosis (BCP). In medicine estimation of the probability of cancer relapse of a patient can help planning the patient's treatment. Here, the authors applied two different imputation methods (mean and hotdeck) to tackle the problem of missing values in data on breast cancer operated patients. According to the authors in [14], the use of imputation significantly improves the prediction when the data are grouped according to risk of cancer relapse. Missing values are also common in meteorological data such as air pollution data and observations on atmospheric parameters. For such data sets, there is no way to regenerate the data by repeating experiments, as can be done in some other application domains. In such applications, the use of appropriate imputation techniques seems the only way to make an effective use of the data with missing values. In [15] authors successfully used multiview methods for imputation of missing values in air quality data by taking into account spatial, temporal, global, and local views.

There have been several attempts to use auto-encoders for dealing with missing data. All most all authors used auto-encoders with a bottleneck layer for handling of missing data [16–24]. Use of a bottleneck layer is useful for feature extraction and data compression. In our view, for finding missing values, the philosophy should be a bit different. Here the primary objective should be better reproducibility of the inputs. Consequently, instead of a bottleneck layer, we have used a latent space of dimension much higher than that of the input. We use a two-stage training scheme which makes an innovative use of the training data without missing values in such a manner that the network is better equipped to handle missing values.

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.07.009.

* Corresponding author.
E-mail addresses: cshuvrajyoti@gmail.com (S.J. Choudhury), nrpal59@gmail.com (N.R. Pal).

https://doi.org/10.1016/j.knosys.2019.07.009
0950-7051/

Here our objective is to design a classifier using a training set with missing values, which is often needed for different practical problems, such as use of gene expression data on cancer patients to design a diagnostic prediction system. For this, we use the trained auto-encoder to predict missing values based on the hypothesis that a good choice for a missing value would be the one which can reconstruct itself via the auto-encoder. Like [19] this idea helps us to define an objective function whose minimization results in the missing values. Since we use gradient descent to minimize the objective function, the initial choice of the missing value is important. For this we use a simple but judiciously chosen initial value. Once the missing value is estimated we train a classifier using the union of the data without missing value and the data which are made complete using our network. We also train another classifier of the same type with the same configuration using the corresponding complete dataset. After that, we use a complete test data set to test both classifiers and report the difference between the accuracies of these two systems. We tested the effectiveness of the proposed algorithm on fourteen challenging datasets and compared the results of the proposed algorithm with eight state-of-art imputation methods with respect to classification accuracy.

The remaining part of the article is organized as follows. Sections 2 and 3 describe the literature review and the proposed method. Experimental results and analysis are demonstrated in Section 4. Section 5 concludes the paper.

2. Literature review

Authors in [1] categorized imputation techniques into two broad groups: statistical imputation techniques and machine learning-based imputation technique. The statistical imputation techniques use common statistical approaches such as the replacement of missing values by mean of the available data and use of regression models [25–27]. Sometimes the missing values are replaced by zero in zero imputation. For a regression-based approach, the complete data points are used to find a regression model to predict the missing values. If k features have missing values, then typically k-regression models are used.

According to [28] the performance of an imputation method depends on the size of the dataset and missing values present in them. Here, the authors analyzed of the performance of four imputation packages in R on two datasets using three predictive algorithms-Logistic Regression, Support Vector Machine and Artificial Neural Network. The authors measure the performance of the packages in terms of imputation time, imputation efficiency and effect on the variance. The authors also show that the performance of an imputation method depends on the predictive algorithm which is a function of dataset characteristics. There are different ways of handling missing data [25–27,29–32]. The most simple approach is to use only the subset that is complete; in other words, such a method ignores all feature vectors with missing values and the remaining subset is used for analysis. Although very simple, such an approach will be useful only when a small fraction data points have missing values. A more useful and popular approach is to impute the missing values using different techniques and then use the data set consisting of both feature vectors without missing values and the feature vectors with imputed missing values. These techniques often depend on the problem that one likes to solve using the data set. For example, to cluster an incomplete data set, two types of frameworks may be adopted: single view clustering and multi-view clustering. The single view clustering [33–41] is the traditional one, whereas multi-view clustering [42–47] is a comparatively new technique to handle missing data. In single view clustering-based approaches, a datum is represented with all of its features whereas, in multi-view clustering-based approaches, a datum is represented using multiple subsets of features. Recently, in [15] authors proposed a multi-view learning-based method to fill missing values for geo-sensory time series data simultaneously considering spatial, temporal, global and local views. The method takes into account both the temporal correlation between observations in a time series and spatial correlation between different time series.

Yu et al. [48] proposed an interesting iterative method for estimation of missing values in Microarray data. It uses a locally auto-weighted least square method, where for a target gene with missing values, the algorithm assigns importance (weights) to the neighboring genes. In the cold and hot deck imputation [49] method, a missing value is imputed using a data point which is the closest in terms of the available features. To account for the uncertainty associated with missing values, a multiple-imputations method imputes a missing value by a set of plausible ones and thereby creating multiple datasets [50]. This procedure involves three distinct phases: 1. The missing values are filled multiple times by a set of plausible ones and thereby creating multiple datasets; 2. Each dataset is considered a complete dataset and analyzed by using standard procedures; 3. The results from all datasets are combined for the inference. There are several other machine learning procedures to impute missing data. For example, k-Nearest Neighbor (KNN) [29] is a simple and useful method to impute a missing value using the nearest neighbor where the distance is computed using only the available features.

As mentioned earlier, there have been several attempts to use auto-encoders (AEs) for handling missing values [16–24]. One of the early work on using AE for handing of missing value is by Chung and Merat [16]. They use an AE with a bottleneck layer for the latent space representation. The objective of this work is to detect sensor failure and recover the missing values. For this, to the extent we understand from the paper, they added a random Gaussian noise to raw senor readings and the output of the AE is taken as the recovered value. In [17] authors train an AE using the training data. After this when an instance with missing values appear, they initialize the missing values with zeros and the instance with the imputed values is applied as an input to the AE. The reconstructed output of the AE is then fed back as an input to the AE. This process is repeated till convergence. Clearly if the AE is a contractive function then the iteration will converge to a fixed point irrespective of the initial values. If the AE mapping is nonexpansive, then the final value will depend on the initial condition. Consequently, they suggest to use some search method to find the missing values so that input and output of the AE match to a pre-specified degree. They neither provide any specific search method nor any results. In [18] authors make an application of AE-based missing value recovery for a jet aircraft engine. Here authors mention "The output of the auto-encoder is subtracted from the distance to specify the error". It is not clear what exactly defines the error. This error is then used to incrementally and iteratively estimate the missing values. It is not mentioned how the missing values were initialized and what search technique was used. Thompson et al. in [19] discussed the contractive nature of AEs and inferred that AEs are *nearly nonexpansive* for most data. Here they proposed two objective functions to find the missing values. The first one suggested to minimize the square error between the input and output considering only the missing features while the second one considered minimization of the square error considering all features. The two objective functions, respectively, can be viewed as relating to unconstrained and constrained search processes. The unconstrained search tries to reduce the auto-encoding error of the missing values only while the constrained search finds the missing values minimizing the total auto-encoding error. Authors in [20] proposed a robust auto-associative neural network, which is essentially the denoising AE

and used the same for online monitoring of a simulated BWR component. Here unlike our approach, while training randomly selected feature values are corrupted by random noises. These authors also use an AE with a bottleneck layer and they estimate the missing value by passing the missing value twice (or more) through the AE. In this paper based on simulated data they discussed several applications. For example, in order to classify accidental events, they projected the data in two dimension using an AE and the classification was done in the projected space based on the nearest neighbor rule using Euclidean distance. Marwala and Chakraverty [21], on the other hand, used AEs with different number of hidden nodes and used genetic algorithms for finding the set of missing values. Unlike our method, their objective was to use the imputed data for fault classification. In [22] to minimize the auto-encoding error, authors use the Particle Swarm Optimization (PSO) method. For such a method [22] one needs to choose several parameters including the number of particles. Later in [23] and [24] authors use Evolutionary Particle Swarm Optimization (EPSO) for minimizing the auto-encoding errors. Here, several local autoencoders are used, where each is trained using a particular cluster of data. These methods discussed in [17–24], train the network using only the complete training data or sometimes training data corrupted by random noise. But we first train the network with the complete data and then create data with missing values from the complete data and refine the AE using both complete and incomplete data. This helps the AE to equip itself to deal with the missing values. We use gradient decent to estimate the missing values and since the output of the gradient search is depends on the initial condition for non-convex functions, we use a simple but judiciously chosen method for initialization of the missing values. More importantly, almost all of these methods use a bottleneck layer as the latent space, while we use an encoding layer of size much bigger than that of input with two purposes: First to ensure a better reconstruction of the input and second the data in the projected space of higher dimension are likely to be flattened [51]. In [51] authors remarked that unfolding of class manifolds using deep networks results in higher dimension of the data which makes the classes more separable. Moreover, like these methods, our objective is not just the restoration of missing values, but how we can impute missing values so that we can use data with missing values in designing more effective classifiers.

The self organizing map (SOM) [30,31] and multilayer perceptron (MLP) [32] are the other two machine learning techniques that have also been used to impute missing values. There are several other machine learning techniques to impute missing data [52–56]. A knowledge-based method has also been developed that can deal with missing values which is non-numeric in nature [57]. Chih-Fong et al. [58] proposed a modified form of the mean imputation method. First, the dataset is divided into complete and incomplete datasets and the complete dataset is divided into individual classes. Then the mean and standard deviation of each feature is computed. The authors find the distance of each data point in a class from the class center and the median distance is used as a threshold. Now for a data point with one missing value from the class i, the missing value is replaced by the mean value of that feature for class i. If the distance of the imputed feature vector from the class mean is less than the threshold, then the imputation is accepted. Otherwise, two imputed vectors are created: one with mean + standard deviation and other with the mean - standard deviation of that feature for class i.

After imputing incomplete data, there are various measures to check how properly the missing values are imputed. Some popular measures such as finding the root-mean-squared error (RMSE) [59] and index of agreement (d_2) [60] between the original and the imputed data can be done only when the ground truths are known. In some cases, after finding missing values authors use a classifier to classify the data with the imputed values [1] while in some other cases the imputed missing data are used to train classifiers. In [56] missing values are imputed using Attribute-based Decision Graphs. Then they design two classifiers of the same type using the imputed and the original data. After that, the authors use a complete test data set to test both the classifiers and report the difference between the accuracies of these two classification systems. Like [56], our objective is also to provide a scheme on how to use data with missing values to design more effective classifiers.

3. Proposed method

In this section, the proposed method is presented. Here first we discuss the formulation of our method. Then we discuss the procedure to train a neural network for imputation. Next, we explain how this network can be used. Since we use an autoencoder, we first briefly discuss that.

3.1. Multilayer perceptrons and autoencoders

An autoencoder can be realized using different networks such as the Restricted Boltzmann Machines, Radial Basis Function networks and MLPs. Here we use an MLP. To ease our discussion we consider one hidden layer here but an extension to multiple hidden layers is straightforward. Let, p be the number of features, q be the number of nodes in the hidden layer, $\mathcal{S}$ be the activation function at any node of this network (note that, different layers may use different types of activation functions) and $\mathbf{x}_k$ be the kth data point. Let $\mathbf{t}_k$ be the target output corresponding to $\mathbf{x}_k$, $\mathbf{t}_k = (t_{k1}, \ldots, t_{kr})^T$ where t_{kj} is the jth target output for $\mathbf{x}_k$ and n is the total number of data points. In case of a classifier, suppose there are r classes. Then $\mathbf{t}_k \in R^r$, is the class label vector, while for an autoencoder, $r = p$ and $\mathbf{t}_k = \mathbf{x}_k$, so $t_{kj} = x_{kj} \forall j = 1, \ldots, p$. In the following we adopt the description used in [61]. The input layer is a fanout layer so its input and output are the same.

Every hidden layer node, makes a weighted sum of its inputs and apply a sigmoidal-type function as,

$$\mathcal{S}(z_{kh}) = \frac{1}{1+e^{-z_{kh}}}, \text{ where } z_{kh} = \sum_{i=0}^{p} w_{ih}\mathcal{S}(x_{ki}), h = 1, \ldots, q \qquad \mathcal{S}(z_{k0}) = 1, \forall k. \tag{1}$$

Here w_{ih} is the connecting weight between the ith input node and the hth hidden node, w_{0h} is the bias of the hth hidden neuron and $\mathcal{S}(z_0)$ is the activation function of the biased neuron of the hidden layer.

For the input $\mathbf{x}_k$, the jth node of the output layer computes

$$\mathcal{S}(y_{kj}) = \frac{1}{1+e^{-y_{kj}}}, \text{ where } y_{kj} = \sum_{h=0}^{q} w_{hj}\mathcal{S}(z_{kh}), j = 1, \ldots, p. \tag{2}$$

where w_{hj} is the weight connecting the hth hidden node to the jth output node and w_{0j} is the bias of the jth output neuron. Thus, the system error for the kth training pattern can be computed as follows:

$$E^k = \frac{1}{2}\sum_{j=1}^{p}\left(x_{kj} - \mathcal{S}(y_{kj})\right)^2 + \eta\sum_{ij} W_{ij}^2. \tag{3}$$

where x_{kj} is the jth feature value of the kth training data point, and $(\eta\sum_{ij} W_{ij}^2)$ represents a weight decay term, which prevents overfitting. $\eta > 0$ is an attenuation coefficient.

Algorithm 1 : Training of the proposed method

BEGIN
Input: X.
1: Set N_1, N_2 and n_c.
2: Randomly divide X into $X_1, \cdots, X_{10}$ such that $X = \cup_{i=1}^{10} X_i, X_i \cap_{i \neq j} X_j = \phi$.
3: **for** i=1 to 10 **do**
4: Set $X_{TR_i} = X - X_i$.
5: Set $\hat{V} = \phi$, $\widetilde{V} = \phi$.
6: **for** j=1 to r **do**
7: Set $\hat{X}_j = \{\mathbf{x}_k | \mathbf{x}_k \in$ the j^{th} class$\}$.
8: Find $\widetilde{\mathbf{v}}_j$ by Eq. (4)
9: $\widetilde{V} = \widetilde{V} \cup \{\widetilde{\mathbf{v}}_j\}$.
10: Using K-Means clustering cluster $\hat{X}_j$ into n_c number of clusters and generate n_c new cluster centers $\left\{\hat{\mathbf{v}}_{j,i}, \cdots, \hat{\mathbf{v}}_{j,n_c}\right\}; \hat{\mathbf{v}}_{j,k} \in R^p$.
11: $\hat{V} = \hat{V} \cup \left\{\hat{\mathbf{v}}_{j,1}, \cdots, \hat{\mathbf{v}}_{j,n_c}\right\}$.
12: **end for**
13: Set $V = \hat{V} \cup \widetilde{V}$.
14: **for** k=1 to p **do**
15: To generate X_i^k, take every $\mathbf{x}_j \in X_{TR_i}$, and replace x_{jk} by v_{lk} if $l = argmin_k \left\{||\mathbf{x}_j - \mathbf{v}_k||_*^2; \mathbf{v}_k \in V\right\}$ where $||.||_*$ computed using all but the k^{th} feature.
16: **end for**
17: We train an auto-encoder using X_{TR_i} for N_1 epoches.
18: Retrain the same network for N_2 epoches with the data set $X^{Total} = X_{TR_i} \cup_{k=1}^{p} X_i^k$. For any $\mathbf{x}_j^k \in X_i^k$, the target vector is taken as $\mathbf{x}_j \in X_{TR_i}$.
19: Store the weights of the network into net.
20: $Network_i = net$.
21: **end for**
END

3.2. Training of the network

Here, we are dealing with a classification problem. So, for every training data point $\mathbf{x}_k \in R^p$, there is an associated class label c_k, $c_k = l$, $l \in \{1, \ldots, r\}$. Let, $X \subseteq R^p$ be the training data. We randomly divide X into ten equal parts (equal to the extent possible): $X = \cup_{i=1}^{10} X_i, X_i \cap_{i \neq j} X_j = \phi$. We take $X_{TR_i} = X - X_i, (\forall i = 1, \ldots, 10)$ and $X_{TE_i} = X_i$. In a 10−fold cross validation framework X_{TR_i} will be used for the training and X_{TE_i} will be used for testing, $i = 1, \ldots, 10$. In X_{TR_i} there are data from all r classes. $X_{TR_i} = \cup_{i=1}^{r} \hat{X}_i$, where $\hat{X}_i = \{\mathbf{x}_k | \mathbf{x}_k \in$ the i^{th} class$\}$. Let, the cardinality of $\hat{X}_i$ be n_i, i.e. $n_i = |\hat{X}_i|$. Now we find out the ith class center $\widetilde{\mathbf{v}}_i$ as,

$$\widetilde{\mathbf{v}}_i = \frac{1}{n_i} \sum_{\mathbf{x}_j \in \hat{X}_i} \mathbf{x}_j. \tag{4}$$

In this way we produce r class centers $\widetilde{V} = \{\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, \ldots, \widetilde{\mathbf{v}}_r\}; \widetilde{\mathbf{v}}_i \in R^p$. Now we take the ith class data points, i.e., $\hat{X}_i$ and cluster $\hat{X}_i$ into n_c clusters using a clustering algorithm, say the k-means algorithm. In this way we produce $(n_c \times r)$ cluster centers $\hat{V} = \left\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \ldots, \hat{\mathbf{v}}_{(n_c \times r)}\right\}; \hat{\mathbf{v}}_i \in R^p$. Then by combining $\hat{V}$ and $\widetilde{V}$ we get the total set of centers as, $V = \hat{V} \cup \widetilde{V}$. In this way we produce $(r + n_c \times r)$ cluster centers $V = \left\{\mathbf{v}_1, \ldots, \mathbf{v}_{(r+n_c \times r)}\right\}; \mathbf{v}_i \in R^p$. Now from X_{TR_i} we generate p modified data sets X_i^k, $k = 1, 2, \ldots, p$ as follows. To generate X_i^k for every $\mathbf{x}_j \in X_{TR_i}$, we replace x_{jk} by v_{lk} if $l = argmin_k \left\{\|\mathbf{x}_j - \mathbf{v}_k\|_*^2\right\}$.

This means that we assume that the kth feature is missing and the kth feature value is imputed by the kth feature value of the centroid which is closest to $\mathbf{x}_i$ in terms of distance measure $\|.\|_*$ computed using all but the kth feature.

Now we train an auto-encoder using X_{TR_i} for N_1 epoches. Next we retrain the same network for N_2 epoches with the data set $X^{Total} = X_{TR_i} \cup_{k=1}^{p} X_i^k$. For any $\mathbf{x}_j \in X_{TR_i}$ as well as any $\mathbf{x}_j^k \in X_i^k$, the target vector is taken as $\mathbf{x}_j \in X_{TR_i}$.

The proposed training method is provided in Algorithm 1.

3.3. Imputation and testing procedure

After training the autoencoder, we want to impute missing values using our scheme and use $X_{TE_i}; i = 1, \ldots, 10$ to test how effective the imputation method is. Before testing first we take $X_{TR_i}; i = 1, \ldots, 10$ and randomly delete data values by varying the rates of the missing attribute values (5%, 10%, 20%, 30%, 40%, 50%) to create X'_i. If $\mathbf{x}_j \in X'_i$, we initialize the missing values of $\mathbf{x}_j$ using some $\mathbf{v}_l \in V$ as follows. If the kth attribute of $\mathbf{x}_j$ is missing than x_{jk} is imputed by v_{lk} if $l = argmin_m \left\{\|\mathbf{x}_j - \mathbf{v}_m\|_*^2; \mathbf{v}_m \in V\right\}$ where $\|.\|_*$ is computed as mentioned earlier, using all but the missing feature value of $\mathbf{x}_j$. Now to find the optimal missing value of $\mathbf{x}_j$ we minimize auto-encoding error of $\mathbf{x}_j$ and modify $\mathbf{x}_j$ as follows.

The error function E^j for an input data point $\mathbf{x}_j$ is:

$$E^j = \frac{1}{2} \sum_{l=1}^{p} \left(x_{jl} - \mathcal{S}\left(\sum_{h=0}^{q} w_{hl} \mathcal{S}\left(\sum_{k=0}^{p} w_{kh} \cdot x_{jk} \right) \right) \right)^2. \tag{5}$$

We fixed the weights of the network and try to minimize E^j with respect to x_{jk} by modifying the imputed values of x_{jk} as,

$$x_{jk} = x_{jk} - \gamma \frac{\partial E^j}{\partial x_{jk}}. \tag{6}$$

where $\gamma > 0$ is the learning rate. If the autoencoder has modeled the nature of the training data adequately, then a good estimate of the missing value would be that value which can reconstruct itself via the autoencoder.

Thus, given a trained autoencoder, we want an estimate of the missing value that will lead to minimum reconstruction error. After imputation, from the training set X_{TR_i} we get another set X_i^U, which is X_{TR_i} where $k\%$ ($k = 5$, 10, 20, 30, 40 or 50) of missing values have been imputed by the proposed imputation method. Here, $|X_{TR_i}| = |X_i^U|$. After imputing the missing values we train a classifier using X_i^U. We also train another classifier of the same type, with the same configuration parameters using the corresponding original complete dataset X_{TR_i}. Then we use $X_{TE_i}; i = 1, \ldots, 10$, to test both the classifiers and report the difference between the accuracies of these systems. The proposed imputation and testing procedure are provided in Algorithm 2.

4. Experiments

We present the descriptions of data sets, experimental design, results and statistical significance test in this section.

4.1. Data sets

As in [56], in our proposed algorithm it is assumed that only the training set has missing attribute values. To perform the comparisons we consider 14 datasets taken from the UCI repository [62] with no missing attribute values and we divide the experiments into six different sets, by varying the rate of missing values (5%, 10%, 20%, 30%, 40%, 50%). The details of these 14 datasets are summarized in Table 1.

4.2. Experimental set up

To impute the incomplete data we train an autoencoder with one hidden layer and with different number of hidden nodes. We made some adhoc experiments by varying the number of hidden nodes just to get some idea about the desirable number of hidden nodes. We could not do a systematic cross-validation experiments to choose it due to lack of computational resources. Based on our experiments we decide to use $(10 \times p)$ nodes (where p is the

Algorithm 2 : Imputation and testing procedure

BEGIN
Input: $X_{TR_i}, X_{TE_i}, Network_i : i = 1, 2, \cdots, 10; V; N_3;\ \gamma$

Set $Accuracy_d = 0$.
for i=1 to 10 **do**
  Take X_{TR_i} and randomly delete features by varying the rates of the missing values (5%, 10%, 20%, 30%, 40%, 50%) to create X_i'.
  trn= no of samples in X_{TR_i}.
  for k=1 to trn **do**
    for j=1 to p **do**
      if x'_{kj} is missing **then**
        Impute x'_{kj}=v_{lj} if $l = argmin_m \left\{||\mathbf{x}'_k - \mathbf{v}_m||^2_*; \mathbf{v}_m \in V\right\}$ where $||.||_*$ computed using all but the missing feature of $\mathbf{x}'_k$.
      end if
    end for
  end for
  For each data point $\mathbf{x}_j$ with missing value set $E^j = \frac{1}{2}\sum_{l=1}^{p}\left(x_{jl} - \mathcal{S}\left(\sum_{h=0}^{q} w_{hl}\mathcal{S}\left(\sum_{k=0}^{p} w_{kh}\cdot x_{jk}\right)\right)\right)^2$.
  for j=1 to N_3 **do**
    for k=1 to trn **do**
      for l=1 to p **do**
        if x'_{kl} is missing **then**
          $x'_{kl} = x'_{kl} - \gamma\frac{\partial E^k}{\partial x'_{kl}}$.
        end if
      end for
    end for
  end for
  Set $X_i^U = X'$.
  Train a classifier using X_i^U.
  Train another classifier of the same type, with the same configuration using X_{TR_i}.
  Use X_{TE_i} to test both the classifiers and store the difference between the accuracies of these systems in dif.
  $Accuracy_d = Accuracy_d + dif$
end for
$Accuracy_d = Accuracy_d/10$.
Report $Accuracy_d$.
END

Table 1
Details of the 14 UCI data sets.

Dataset	# Features	# Classes	# Instances
Iris	4	3	150
Wine	13	3	178
Balance	4	3	625
Sonar	60	2	208
Glass	10	7	214
Pima	8	2	768
WDBC	30	2	569
WPBC	34	2	198
Waveform	21	3	5000
Heartspectf	44	2	80
Haberman	3	2	306
Heart	13	2	270
Vehicle	18	4	846
Liver	6	2	345

number of input features) in the hidden layer. To train a network for each data set we take the learning rate $\eta = 0.9$ and $n_c = 2$ (No of clusters). To impute incomplete data we take the learning coefficient $\gamma = 0.005$ and the number of iterations that is used to refine the imputed value is $N_3 = 10\,000$. To reduce experimental overhead we vary the values of N_1 and N_2 for different data sets. If there are p features, then the training set is increased by p times. Since, $(n \times p)$ of Sonar, WDBC, WPBC and Waveform is larger than those others we choose $N_1 = N_2 = 1000$ for these datasets. For the remaining 10 datasets we choose $N_1 = N_2 = 10\,000$. While training an auto-encoder first we train using X_{TR_i}. Then we randomly dropped attribute values by 5%, 10%, 20%, 30%, 40% or 50% from X_{TR_i} and use the trained auto-encoder to impute the missing attribute values to obtain X_i^U. Then we use X_i^U to train a classifier. We train another classifier with the same configuration using the original attribute values of X_{TR_i}. Now we pass the remaining X_{TE_i} to both classifiers for testing and we calculate the difference between the accuracies of the two classifiers. In particular, we compute (accuracy of the classifier trained with X_{TR_i} − accuracy of the classifier trained with X_i^U). For each of the 14 datasets, we repeat the experiment for 10 different weight initializations, for each of the 10 folds and take the difference of the average accuracies. We train eight different classifiers. All simulations have been done on a desktop with Intel(R) Core(TM) i7 CPU 3.40 GHz, 32 GB RAM, Windows 8.1 operating system. We have used Matlab R2015b for implementation of the entire algorithm.

4.3. Results and discussions

We compare the proposed method with eight state-of-the-art imputation techniques (AbDG, CAbDG, CART, MEAN, Norm, F. Imp., L. Reg. and R. F.) using fourteen datasets and eight different classifiers. The classifiers are Support Vector Machine (SVM), Multinomial Logistic Regression (MUL), Naive Bayes (NBA), Multilayer Perceptron (MLP), Parzen classifier (PRZ), k-Nearest Neighbor (KNN), Classification And Regression Tree (CAR), and Probabilistic Neural Networks (PNN). To implement these eight classifiers on the imputed data set produced by our method we use Matlab toolbox and we choose parameters as described in [56]. For comparing our methods with other imputation methods we use the results described in [56].

We compare the performance of our method with that of others by using both figures and tables. In Fig. 1 we display the difference in accuracy between the classifiers trained with imputed data and without missing values (original data) averaged over all datasets, for each missing attribute rate. Lower the magnitude of the difference better the performance of the missing value handling methods. From Fig. 1 we see that at 5%, 10% and 20% missing rates, the performance of our method is marginally inferior or equal to that of other methods. But at higher missing rates (30%, 40% and 50%), the performance of our method is noticeably better than that of other methods.

We now compare the performance of our method with other methods with respect to the different classifiers at different rates of missing percentages. From Fig. 2 it is seen that when the missing rate is 5%, among the 8 classifiers our method performs better on the imputed data using MUL and CAR classifiers. Whereas for the AbDG imputation method, 4 classifiers (MUL, MLP, PRZ, PNN) perform better. Similarly, for CAbDG, 5 classifiers (MUL, NBA, MLP, PRZ, PNN), for Mean, 2 classifiers (MUL and PNN), for each of Norm, L.Reg, and R.F., only one classifier performs better on the imputed data.

When the missing rate is 10% among 8 classifiers none of the imputation methods gave a consistent performance on multiple classifiers. When the missing rate is 20% among the 8 classifiers our method performs better on the imputed data using the MUL classifier. In case of other methods for Norm and L.Reg, only the NBA classifier performs better on the imputed data.

Similarly, when the missing rates are 30% and 40% among the 8 classifiers our method performs better on the imputed data using 1 classifier (MLP and MUL, respectively) and for Norm and L.Reg, only the NBA classifier performs better on the imputed data.

But, when the missing rate is 50% from all methods only our method performs better on the imputed data using the MUL classifier. In 50% missing rate of the 8 classifiers, our method performs better than others using 7 classifiers. Only with the PNN classifier, Mean method performs better than our method. So, in summary, we can say that from 20% missing rate onwards the

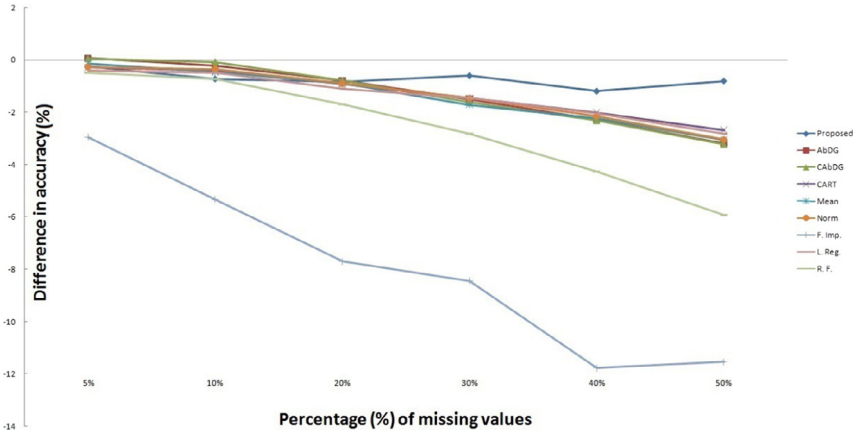

Fig. 1. Differences in accuracy between classifiers trained with imputed data and classifier trained without missing values averaged over all data sets for each result, for each missing attribute rate and imputation method.

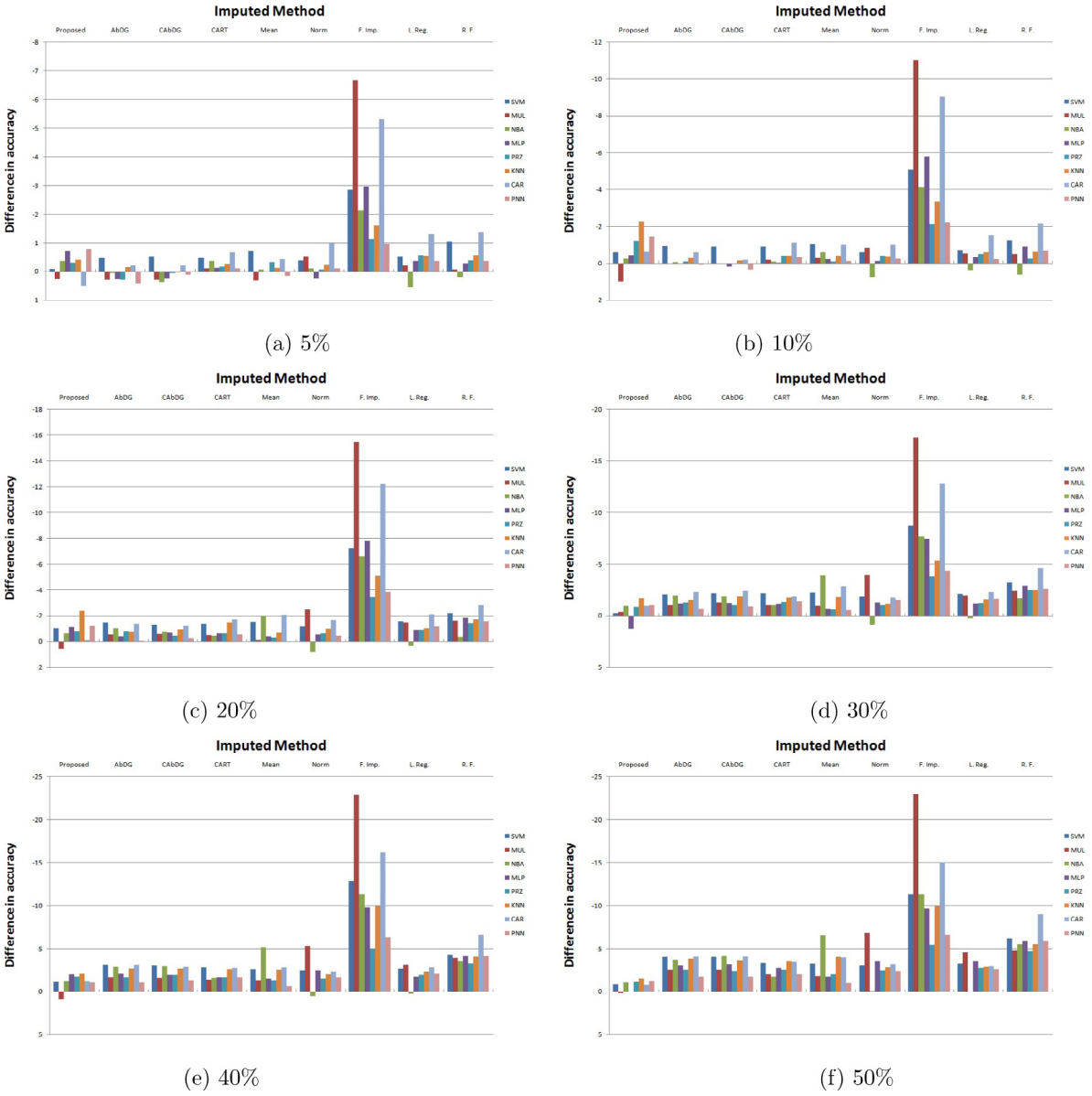

Fig. 2. Differences in accuracy between imputed data and original data averaged by classifiers for the imputation methods under comparison for different classifiers averaged over different datasets (In vertical axis Accuracy difference (%) between imputed data and original data averaged over the 14 domains and in horizontal axis different imputation methods are presented, larger the magnitude of negative values, better in the performance).

Table 2
Winning methods in different missing rate (Percentage wise).

	Our	AbDG	CAbDG	CART	Mean	Norm	F. Imp.	L. Reg.	R. F.
5%	32.17	19.13	12.17	4.35	14.78	7.83	4.35	4.35	0.87
10%	25.20	17.07	16.26	5.69	8.13	6.50	4.88	9.76	6.50
20%	30.09	15.93	11.50	4.42	18.58	8.85	3.54	5.31	1.77
30%	35.25	11.48	10.66	4.92	13.11	13.93	3.28	4.92	2.46
40%	39.82	7.96	6.19	1.77	19.47	12.39	3.54	8.85	0.00
50%	44.74	5.26	9.65	5.26	14.04	11.40	2.63	4.39	2.63

performance of the classifiers designed using the imputed data by the proposed method is similar to or better than the that of the classifiers designed using the imputed data by other methods.

The performance of our method can be viewed better than that of others, if the classification accuracy of our method is higher than that of others using different classifiers on different datasets. So, keeping this mind in Fig. 3 and Table 2 we show the winning percentage of different imputation methods at different missing rates using different classifiers on different datasets. There are 14 data sets, 8 classifiers and 9 imputation methods including the proposed method. For each pair of dataset and classifier, among the 9 imputation methods, we find the winner considering the difference in accuracies of the two classifiers trained using the original data and the imputed data. This is done for all 14×8 pairs of dataset and classifier. Let W_i be the no of times the ith imputation method becomes the winner. In Table 2 and Fig. 3 we report the percentage of winning computed as $\frac{W_i}{\sum_k W_k} \times 100\%$. From Table 2 and Fig. 3 we see that the winning percentage of our method is the highest among all nine methods at all missing rates. At 5% missing rate our method wins 32.17% times. Next best method AbDG which wins 19.13% time at 5% missing rate. The winning performance of other is comparatively less than these two methods at 5% missing rate. At 10% missing rate the winning percentage of our method decreases from 32.17% to 25.20% but still, it is higher than that of others. Performance of the next best method, AbDG, is also decreased from 19.13% to 17.07%. As the missing rate increases beyond 10% the winning percentage of the proposed method increases. At 20% it is 30.09%, at 30% it is 35.25%, at 40% it is 39.82%, and at 50% it is 44.74%. From others, for AbDG and CAbDG the performance is decreased and the winning percentage for the CART, Mean, Norm, F.Imp, L.Reg, and R.F. fluctuates with the increase of missing rates. In conclusion, we observe that from 20% missing rate onwards the classification performance of our method is significantly better than that of others and increases monotonically.

4.4. Statistical significance: Wilcoxon signed rank test

We have performed Wilcoxon signed rank tests (1-tail) for pairwise comparison of the proposed method with eight imputation methods AbDG, CAbDG, CART, MEAN, Norm, F. Imp., L. Reg. and R. F. at the significance label $\alpha = 0.05$. Here, the null hypothesis (H_0) is that there is no significant difference between the comparing two methods. We tested the effectiveness of the proposed algorithm on fourteen datasets and compared the results of the proposed algorithm with eight imputation techniques with respect to eight classifiers at six different missing rates. In this fashion, there will be 384 ($= (8 \times 8 \times 6)$) pair-wise tests that have been summarized in Table 3. In all tests, the pairs are (L_1, L_j) where L_1 is the proposed method and $L_j, j = 2, \ldots, 9$ are the eight imputation methods (AbDG, CAbDG, CART, MEAN, Norm, F. Imp., L. Reg., and R. F). In Table 3, in the columns $2-4$, the entries (1) $L_1 > L_j$; (2) $L_1 \approx L_j$; and (3) $L_1 < L_j$, respectively, denote that: (1) the performance of L_1 is significantly better than L_j; (2) there is no significant difference between the performances of L_1 and L_j; and (3) the performance of L_1 is significantly worse than L_j. We treat these cases, respectively, as win, draw, and loss of L_1 when compared to L_j. From Table 3 it is seen that at 5% missing rate from 64 tests our method wins in 7 cases, draws in 52 cases and loses in 5 cases. In 10% missing rate the number of loss of our method increases from 5 to 14 and the number of draws with others decreases from 52 to 38, while the number of wins over others increases from 7 to 12. At 20% the number of cases the proposed method lost decreases from 14 to 5, the number of draws with others increases from 38 to 47, but winning frequency remains the same as previous. From 30% onwards the winning frequencies of the proposed method increases with missing rates and these are 22, 25 and 41 at 30%, 40% and 50% respectively. On the other hand, our method lost only 1, 1, 0 times at 30%, 40% and 50%, respectively. In summary, we can say that beyond 10% missing values, with the increase in missing rates the performance of our method improves. Overall in 30.99% cases the performance of the proposed method is significantly better than others, in 6.77% cases it is worse than others and in the remaining 62.24% cases it is comparable to other methods.

Table 3
Wilcoxon Signed Rank Test Result in different missing rate for $\alpha = 0.05$.

Missing rate	$L_1 > L_j$	$L_1 \approx L_j$	$L_1 < L_j$
5%	7	52	5
10%	12	38	14
20%	12	47	5
30%	22	41	1
40%	25	38	1
50%	41	23	0
Total	119	239	26
Percentage	30.99	62.24	6.77

5. Conclusion and discussion

We have proposed a mechanism so that data with missing values can be effectively used to design classifiers and other machine learning algorithms. There are many real-life applications which demands use of data with missing values to design classifiers. For example gene expression data generated by DNA-Microarray technology usually suffer from missing values due to different reasons such as experimental error and hybridization failure. For most applications of gene expression data the number of instances available is very limited. Often repetition of the experiment is not possible and even if the experiment is repeated, there is no guarantee that there will not be any missing values (Usually some values will be missing, which may be different from the previous missing set). In such a case, we can use the proposed method to impute the values and design a better diagnostic prediction system. We can also use the data imputed by the proposed method for further analysis such as identification of bio-markers and discovery of genes regulatory networks.

Typically, an auto-encoder is used for feature extraction or data compression, but here we use AEs for modeling the characteristics of a given dataset. To realize a better reconstruction of the data, instead of a bottleneck layer, we use a latent space of much higher dimension than that of the input. The AE is trained using a two-phase training scheme: First we use the complete data to train the AE. The AE is further retrained using an augmented data set so that the AE is better equipped to deal with missing values. Now to use a data set with missing values for classifier design, for each instance we impute the missing value using a simple yet effective technique (just an initial guess) and then modify it minimizing the reconstruction error using an all-ready trained AE network. We use the union of data with imputed values and the remaining complete data to train a classifier. For

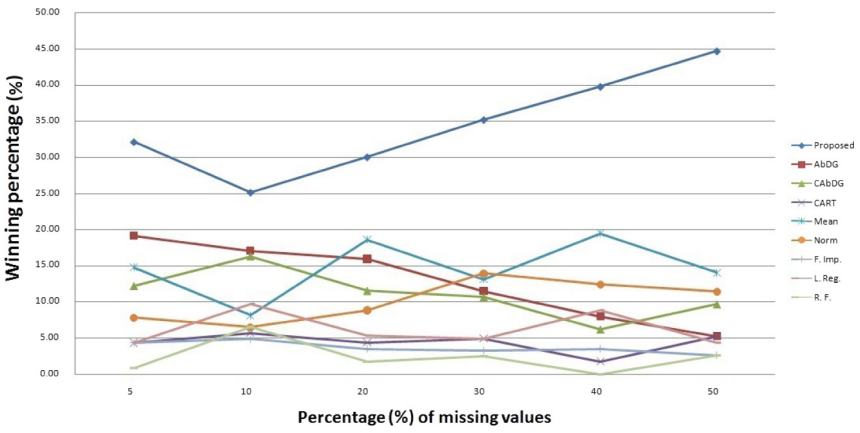

Fig. 3. Winning percentage overall classifiers over all datasets of different imputation methods in different missing rate.

comparison, we use the original complete data to train a similar classifier. Then we pass a test set through both the classifiers and report the difference in accuracies. This procedure is also followed by others [56]. We tested the effectiveness of the proposed algorithm on fourteen challenging datasets (downloaded from University of California at Irvine (UCI) repository) and compared the results of the proposed algorithm with eight state-of-art imputation techniques using eight classifiers with six different missing rates. We have shown that for higher missing rates the performance of our method is much better than several state-of-the-art methods. The performance of our method may depend on the initial guess for the missing values. It would be worth to explore the use of other methods to make the initial guess for the missing values. For our method the size of the augmented training set is $(p+1)\times n$ which may demand significant computational resources for large n. The computational complexity increases linearly with n. Note that, we cannot reduce the overhead due to large p as any feature may be missing. However, we can reduce the overhead due to large n using different schemes. For example, from each class we can select a small percentage of random instances to generate the augmented training data. As an example, if we use 25% of the samples, then the size of the augmented training data would be $n+(\frac{p\times n}{4}) = n(1+\frac{p}{4})$. This will reduce computation significantly and is not likely to change the performance significantly. We can also use more sophisticated methods such as a self-organizing map to quantize the data and use only a few instances from the set of instances that are mapped to each SOM prototype. For example, if SOM has k prototypes then the n training points will be divided into k subsets and from each such subset we can choose one (the one closest to the prototype) or two (the closest one and the furthest one, because considering two very similar data points will not add new information). In future, we like to explore some of these ideas.

References

[1] P.J. García-Laencina, J.-L. Sancho-Gómez, A.R. Figueiras-Vidal, Pattern classification with missing data: a review, Neural Comput. Appl. 19 (2) (2010) 263–282.
[2] L.N. Nguyen, W.T. Scherer, Imputation Techniques to Account for Missing Data in Support of Intelligent Transportation Systems Applications, Tech. Rep., 2003.
[3] K. Lakshminarayan, S.A. Harp, T. Samad, Imputation of missing data in industrial databases, Appl. Intell. 11 (3) (1999) 259–275.
[4] C. Ji, A. Elwalid, Measurement-based network monitoring: missing data formulation and scalability analysis, in: Information Theory, 2000. Proceedings. IEEE International Symposium on, IEEE, 2000, p. 78.
[5] M.H. Le Gruenwald, Estimating missing values in related sensor data streams, in: COMAD, 2005.
[6] H.S. Mohammed, N. Stepenosky, R. Polikar, An ensemble technique to handle missing data from sensors, in: Sensors Applications Symposium, 2006. Proceedings of the 2006 IEEE, IEEE, 2006, pp. 101–105.
[7] M. Cooke, P. Green, M. Crawford, Handling missing data in speech recognition, in: Third International Conference on Spoken Language Processing, 1994, pp. 1555–1558.
[8] S. Parveen, P. Green, Speech enhancement with missing data techniques using recurrent neural networks, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings, Vol. 1, (ICASSP'04), IEEE, 2004, pp. 733–736.
[9] P. Kofman, I.G. Sharpe, Using multiple imputation in the analysis of incomplete observations in finance, J. Financ. Econ. 1 (2) (2003) 216–249.
[10] G. DiCesare, Imputation, Estimation and Missing Data in Finance, University of Waterloo, 2006.
[11] M.A. Proschan, R.P. McMahon, J.H. Shih, S.A. Hunsberger, N.L. Geller, G. Knatterud, J. Wittes, Sensitivity analysis using an imputation method for missing binary data in clinical trials, J. Statist. Plann. Inference 96 (1) (2001) 155–165.
[12] M.K. Markey, A. Patel, Impact of missing data in training artificial neural networks for computer-aided diagnosis, in: Machine Learning and Applications, 2004. Proceedings. 2004 International Conference on, IEEE, 2004, pp. 351–354.
[13] P. Liu, E. El-Darzi, L. Lei, C. Vasilakis, P. Chountas, W. Huang, An analysis of missing data treatment methods and their application to health care dataset, Adv. Data Min. Appl. (2005) 730.
[14] J.M. Jerez, I. Molina, J.L. Subirats, L. Franco, Missing data imputation in breast cancer prognosis, BioMed 6 (2006) 323–328.
[15] X. Yi, Y. Zheng, J. Zhang, T. Li, ST-MVL: Filling missing values in geo-sensory time series data, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, AAAI Press, 2016, pp. 2704–2710.
[16] D. Chung, F.L. Merat, Neural network based sensor array signal processing, in: Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on, IEEE, 1996, pp. 757–764.
[17] S. Narayanan, R.J. Marks, J.L. Vian, J.J. Choi, M.A. El-Sharkawi, B.B. Thompson, Set constraint discovery: missing sensor data restoration using autoassociative regression machines, in: Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on, Vol. 3, IEEE, 2002, pp. 2872–2877.
[18] S. Narayanan, J.L. Vian, J.J. Choi, R.J. Marks, M.A. El-Sharkawi, B.B. Thompson, Missing sensor data restoration for vibration sensors on a jet aircraft engine, in: Proceedings of the International Joint Conference on Neural Networks, 2003, Vol. 4, IEEE, 2003, pp. 3007–3010.
[19] B.B. Thompson, R.J. Marks, M.A. El-Sharkawi, On the contractive nature of autoencoders: Application to missing sensor restoration, in: Proceedings of the International Joint Conference on Neural Networks, 2003, Vol. 4, IEEE, 2003, pp. 3011–3016.
[20] M. Marseguerra, A. Zoia, The autoassociative neural network in signal analysis: II. Application to on-line monitoring of a simulated BWR component, Ann. Nucl. Energy 32 (11) (2005) 1207–1223.
[21] T. Marwala, S. Chakraverty, Fault classification in structures with incomplete measured data using autoassociative neural networks and genetic algorithm, Current Sci. (2006) 542–548.
[22] W. Qiao, Z. Gao, R.G. Harley, G.K. Venayagamoorthy, Robust neuro-identification of nonlinear plants in electric power systems with missing sensor measurements, Eng. Appl. Artif. Intell. 21 (4) (2008) 604–618.
[23] V. Miranda, J. Krstulovic, H. Keko, C. Moreira, J. Pereira, Reconstructing missing data in state estimation with autoencoders, IEEE Trans. Power Syst. 27 (2) (2012) 604–611.

[24] J. Krstulovic, V. Miranda, A.J.A.S. Costa, J. Pereira, Towards an auto-associative topology state estimator, IEEE Trans. Power Syst. 28 (3) (2013) 3311–3318.
[25] J.L. Schafer, Analysis of Incomplete Multivariate Data, Chapman and Hall/CRC, 1997.
[26] S. Menard, Sage University paper series on quantitative applications in the social sciences, in: Applied Logistic Regression Analysis, Sage Publications Thousand Oaks, California, 1995.
[27] R.J.A. Little, D.B. Rubin, Statistical Analysis with Missing Data, Vol. 333, John Wiley & Sons, 2014.
[28] M.L. Yadav, B. Roychoudhury, Handling missing values: A study of popular imputation packages in R, Knowl.-Based Syst. 160 (2018) 104–118.
[29] J.K. Dixon, Pattern recognition with partly missing data, IEEE Trans. Syst. Man Cybern. 9 (10) (1979) 617–621.
[30] T. Samad, S.A. Harp, Self—organization with partial data, Network: Comput. Neural Syst. 3 (2) (1992) 205–212.
[31] F. Fessant, S. Midenet, Self-organising map for data imputation and correction in surveys, Neural Comput. Appl. 10 (4) (2002) 300–310.
[32] L.K. Westin, Missing data and the preprocessing perceptron, Univ., 2004.
[33] R.J. Hathaway, J.C. Bezdek, Fuzzy c-means clustering of incomplete data, IEEE Trans. Syst. Man Cybern. B 31 (5) (2001) 735–744.
[34] K. Honda, H. Ichihashi, Linear fuzzy clustering techniques with missing values and their application to local principal component analysis, IEEE Trans. Fuzzy Syst. 12 (2) (2004) 183–193.
[35] D. Li, H. Gu, L. Zhang, A fuzzy c-means clustering algorithm based on nearest-neighbor intervals for incomplete data, Expert Syst. Appl. 37 (10) (2010) 6942–6947.
[36] A.G. Di Nuovo, Missing data analysis with fuzzy c-means: A study of its application in a psychological scenario, Expert Syst. Appl. 38 (6) (2011) 6793–6797.
[37] D. Li, H. Gu, L. Zhang, A hybrid genetic algorithm–fuzzy c-means approach for incomplete data clustering based on nearest-neighbor intervals, Soft Comput. 17 (10) (2013) 1787–1796.
[38] B. Wang, L. Zhang, L. Zhang, Z. Bing, X. Xu, Missing data imputation by nearest-neighbor trained BP for fuzzy clustering, J. Inf. Comput. Sci. 11 (15) (2014) 5367–5375.
[39] J. Li, S. Song, Y. Zhang, Z. Zhou, Robust K-Median and K-Means clustering algorithms for incomplete data, Math. Probl. Eng. 2016 (2016).
[40] S. Datta, S. Bhattacharjee, S. Das, Clustering with missing features: a penalized dissimilarity measure based approach, Mach. Learn. 107 (12) (2018) 1987–2025.
[41] J. Li, S. Song, Y. Zhang, K. Li, A robust Fuzzy c-means clustering algorithm for incomplete data, in: Intelligent Computing, Networked Control, and their Engineering Applications, Springer, 2017, pp. 3–12.
[42] J. Liu, C. Wang, J. Gao, J. Han, Multi-view clustering via joint nonnegative matrix factorization, in: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, 2013, pp. 252–260.
[43] S.-Y. Li, Y. Jiang, Z.-H. Zhou, Partial multi-view clustering, in: Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014, pp. 1968–1974.
[44] W. Shao, L. He, S.Y. Philip, Multiple incomplete views clustering via weighted nonnegative matrix factorization with $L_{2,1}$ regularization, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2015, pp. 318–334.
[45] H. Zhao, H. Liu, Y. Fu, Incomplete multi-modal visual data grouping, in: IJCAI, 2016, pp. 2392–2398.
[46] Q. Yin, S. Wu, L. Wang, Unified subspace learning for incomplete and unlabeled multi-view data, Pattern Recognit. 67 (2017) 313–327.
[47] L. Zhao, Z. Chen, Y. Yang, Z.J. Wang, V.C.M. Leung, Incomplete multi-view clustering via deep semantic mapping, Neurocomputing 275 (2018) 1053–1062.
[48] Z. Yu, T. Li, S.-J. Horng, Y. Pan, H. Wang, Y. Jing, An iterative locally auto-weighted least squares method for microarray missing value estimation, IEEE Trans. Nanobiosci. 16 (1) (2017) 21–33.
[49] G. Kalton, Compensating for Missing Survey Data, Ann Arbor Michigan University of Michigan Insitute for Social Research Survey Research Center 1983, 1983.
[50] J.S. Murray, et al., Multiple imputation: a review of practical and theoretical findings, Statist. Sci. 33 (2) (2018) 142–159.
[51] P.P. Brahma, D. Wu, Y. She, Why deep learning works: A manifold disentanglement perspective, IEEE Trans. Neural Netw. Learn. Syst. 27 (10) (2016) 1997–2008.
[52] E.-L. Silva-Ramírez, R. Pino-Mejías, M. López-Coello, M.-D. Cubiles-de-la-Vega, Missing value imputation on missing completely at random data using multilayer perceptrons, Neural Netw. 24 (1) (2011) 121–129.
[53] C. Gautam, V. Ravi, Counter propagation auto-associative neural network based data imputation, Inform. Sci. 325 (2015) 288–299.
[54] E.-L. Silva-Ramírez, R. Pino-Mejías, M. López-Coello, Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns, Appl. Soft Comput. 29 (2015) 65–74.
[55] C. Gautam, V. Ravi, Data imputation via evolutionary computation, clustering and a neural network, Neurocomputing 156 (2015) 134–142.
[56] J.R.B. Junior, M. do Carmo Nicoletti, L. Zhao, An embedded imputation method via attribute-based decision graphs, Expert Syst. Appl. 57 (2016) 159–177.
[57] Z. Qi, H. Wang, J. Li, H. Gao, FROG: Inference from knowledge base for missing value imputation, Knowl.-Based Syst. 145 (2018) 77–90.
[58] C.-F. Tsai, M.-L. Li, W.-C. Lin, A class center based approach for missing value imputation, Knowl.-Based Syst. 151 (2018) 124–135.
[59] M.G. Rahman, M.Z. Islam, FIMUS: A framework for imputing missing values using co-appearance, correlation and similarity analysis, Knowl.-Based Syst. 56 (2014) 311–327.
[60] C.J. Willmott, Some comments on the evaluation of model performance, Bull. Am. Meteorol. Soc. 63 (11) (1982) 1309–1313.
[61] S. Kumar, Neural Networks: a Classroom Approach, Tata McGraw-Hill Education, 2004.
[62] M. Lichman, et al., UCI machine learning repository, Irvine, CA, 2013.