Neural Comput & Applic
DOI 10.1007/s00521-009-0295-6

ORIGINAL ARTICLE

Pattern classification with missing data: a review

Pedro J. García-Laencina · José-Luis Sancho-Gómez · Aníbal R. Figueiras-Vidal

Received: 15 June 2007 / Accepted: 19 August 2009
© Springer-Verlag London Limited 2009

Abstract Pattern classification has been successfully applied in many problem domains, such as biometric recognition, document classification or medical diagnosis. Missing or unknown data are a common drawback that pattern recognition techniques need to deal with when solving real-life classification tasks. Machine learning approaches and methods imported from statistical learning theory have been most intensively studied and used in this subject. The aim of this work is to analyze the missing data problem in pattern classification tasks, and to summarize and compare some of the well-known methods used for handling missing values.

Keywords Pattern classification · Missing data · Neural networks · Machine learning

1 Introduction

Pattern classification can be defined as the act of taking in raw data and taking action based on the category of the data [1]. As humans, it is easy for everyone (even children) to recognize letters, objects, colors, numbers, etc. From the early development of computers, researchers tried to implement the ability in machines. Pattern classification is the discipline of building machines to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns [1–5]. In general [4], a classical definition of pattern is an entity that can be represented by a set of properties and variables (feature vector). For example, a pattern could be an audio signal, being the corresponding feature vector its frequency spectral components; a patient, being the feature vector the results of his/her medical tests, or a digitized image for a character recognition problem.

Pattern classification was developed starting from the 1960s. It progressed to a great extent in parallel with the growth of research on knowledge-based systems and artificial neural networks. Increasing computational resources, while enabling faster processing of huge data sets, have also facilitated the research on pattern classification, providing new developments of methodology and applications. This interdisciplinary field has been successfully applied in several scientific areas such as computer science, engineering, statistics, biology, and medicine, among others. These applications include biometrics (personal identification based on several physical attributes such as fingerprints and iris), medical diagnosis (CAD, computer aided diagnosis), financial index prediction, and industrial automation (fault detection in industrial process).

Nowadays, data are generated almost everywhere: sensor networks in Mars, submarines in the deepest ocean, opinion polls about any topic, etc. Many of these real-world applications suffer a common drawback, missing or unknown data (incomplete feature vector). For example, in an industrial experiment some results can be missing because of mechanical/electronic failures during the data acquisition process. In medical diagnosis, some tests cannot be done because either the hospital lacks the necessary medical equipment, or some medical tests may not be

P. J. García-Laencina (✉) · J.-L. Sancho-Gómez
Dpto. Tecnologías de la Información y las Comunicaciones, Universidad Politécnica de Cartagena, Plaza del Hospital 1, 30202 Cartagena (Murcia), Spain
e-mail: pedroj.garcia@upct.es

A. R. Figueiras-Vidal
Dpto. Teoría de Señal y Comunicaciones, Universidad Carlos III de Madrid, Avda. de la Universidad, 30, 28911 Leganés (Madrid), Spain

appropriate for certain patients. In the same context, another example could be an examination by a doctor, who performs different kinds of tests; some test results may be available instantly, others may take several days to complete. Anyway, it might be necessary to reach a preliminary diagnosis instantly, using only those results that are available. In a social survey, respondents may refuse to respond to some questions. The subject of missing data has been treated extensively in the literature of statistical analysis [6–8], and also, but with less effort, in the pattern recognition literature. The ability of handling missing data has become a fundamental requirement for pattern classification, because inappropriate treatment of missing data may cause large errors or false results on classification. In addition, it is a more common problem in real-world data. As an example, when a sensor fails in a production process, it might not be necessary to stop everything if sufficient information is implicitly contained in the remaining sensor data. Another clear example of the importance of handling missing data is that 45% of data sets in the UCI repository have missing values, which is one of most commonly used data set collection for benchmarking machine learning procedures.

In general, pattern classification with missing data concerns two different problems, handling missing values and pattern classification. Most of the approaches in the literature can be grouped into four different types depending on how both problems are solved,

- Deletion of incomplete cases and classifier design using only the complete data portion.
- Imputation or estimation of missing data and learning of the classification problem using the edited set, i.e., complete data portion and incomplete patterns with imputed values.
- Use of model-based procedures, where the data distribution is modeled by means of some procedures, e.g., by expectation–maximization (EM) algorithm.
- Use of machine learning procedures, where missing values are incorporated to the classifier.

In the two-first types of approaches, the two problems, handling missing values (data deletion and imputation, in each case, respectively) and pattern classification, are solved separately; in contrast, the third type of approaches model the probability density function (PDF) of the input data (complete and incomplete cases), which is used to classify using the Bayes decision theory. Finally, in the last kind of approaches, the classifier has been designed for handling incomplete input data without a previous estimation of missing data. Figure 1 resumes the different approaches in pattern classification with missing data.

This work reviews the most important missing data techniques in pattern classification, trying to highlight their advantages and disadvantages. An excellent reference for missing data is the book written by Little and Rubin [6], which gives an accurate mathematical and statistical background in this field. However, this book does not deal with machine learning solutions for missing data. Our main goal is to show the most representative and useful procedures for handling missing data in classification problems, with a special emphasis on solutions based on machine

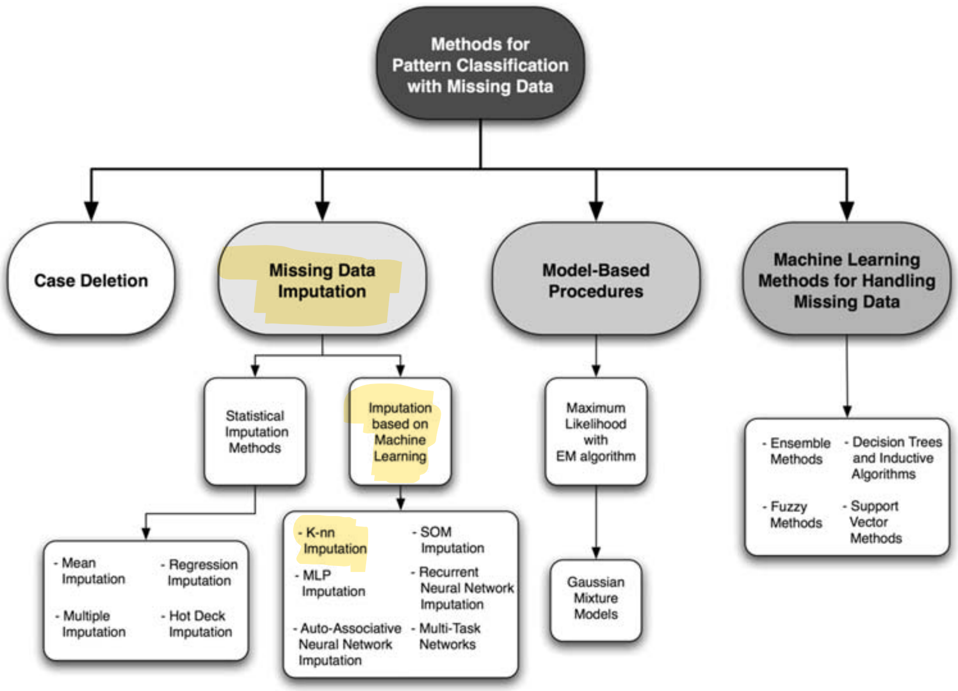

Methods for Pattern Classification with Missing Data
Case Deletion
Missing Data Imputation
Model-Based Procedures
Machine Learning Methods for Handling Missing Data
Statistical Imputation Methods
Imputation based on Machine Learning
Maximum Likelihood with EM algorithm
- Ensemble Methods
- Decision Trees and Inductive Algorithms
- Fuzzy Methods
- Support Vector Methods
- Mean Imputation
- Regression Imputation
- Multiple Imputation
- Hot Deck Imputation
- K-nn Imputation
- SOM Imputation
- MLP Imputation
- Recurrent Neural Network Imputation
- Auto-Associative Neural Network Imputation
- Multi-Task Networks
Gaussian Mixture Models

Fig. 1 Methods for pattern classification with missing data. This scheme shows the different procedures that are analyzed in this work

learning. Due to space constraints, we are not able to provide a complete and detailed study of proposed solutions for incomplete data classification. Thus, this review paper provides a wide and general overview of the state-of-the-art in this field. The remainder of this paper is structured as follows. The fundamental notions of pattern classification with incomplete values are introduced in Sect. 2. Section 3 describes both real applications where missing values and the different missing data mechanisms are presented. From Sect. 4–7, several methods for dealing with missing data in classification tasks are discussed. Section 8 shows obtained experimental results in different decision problems. This section compares some representative and useful imputation approaches in order to evaluate the influence of missing data estimation in the generalization performance obtained by a neural network classifier. Finally, the main conclusions end this paper.

2 Pattern classification with missing data

In pattern classification, a pattern[1] is represented by a vector of d features or attributes (continuous or discrete), i.e., $\mathbf{x} = [x_1, x_2, \ldots, x_i, \ldots, x_d]^T$. In addition, each pattern belongs to one of c possible classes or categories: C_1, C_2, …, C_c. In general, a pattern classification system is operated in two modes [5]: training (learning) and classification (testing). During the training phase, the classification system is completely designed (to set up its internal parameters) using a set of training samples, which is named training set. We consider that training samples are labeled (supervised learning), i.e., the label t of a training pattern $\mathbf{x}$ represents the category to which this pattern belongs. In the testing mode, the trained system assigns the input pattern to one of the classes under consideration based on its attributes. The classifier performance depends on both the number of training samples and the specific values of these samples. At the same time, the goal of designing a classification system is to classify future test samples, which were not used during the training stage. Figure 2a shows a general pattern classification problem, where each training pattern $\mathbf{x}_n$ has d input features (without any missing data) and an output classification target t_n.

In the classification tasks discussed in this paper, input patterns can have some unknown attribute values (i.e., missing values on its features). In addition, we consider that missing values are not always in the same attribute among the given samples (i.e., it is possible that the i-th attribute value of one sample is missing while the same attribute of another example is known). A missing value is

[1] Henceforth, the terms pattern, input vector, case, observation, sample, and example are used as synonyms.

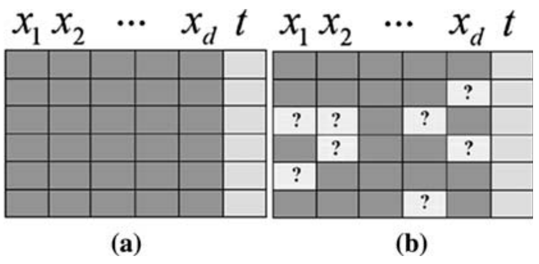


Fig. 2 Two hypothetical pattern classification problems. First, in (**a**), all patterns are completely known (without missing values on its features). Conversely, in (**b**), some input vectors present missing data (with missing values on its features)

referred with '?' symbol; thus, $\mathbf{x_4} = [?, -0.2, ?, 0.3]^T$ means that the fourth training pattern presents missing values at the first and third attributes. Figure 2b shows a classification problem where some input vectors are incomplete.

In order to resume the notation used in this work, let us assume a data set D composed of N labeled incomplete patterns,

$$D = \{\mathbf{X}, \mathbf{T}, \mathbf{M}\} = \{(\mathbf{x}_n, t_n, \mathbf{m}_n)\}_{n=1}^{N} \tag{1}$$

where $\mathbf{x}_n = [x_{1n}, x_{2n}, \ldots, x_{dn}]^T$ is the n-th input vector composed of d features; labeled as $t_n \in [C_1, C_2, \ldots, C_c]$, and $\mathbf{m}_n = [\mathrm{m}_{1n}, m_{2n}, \ldots, m_{dn}]^T$ indicates which input features are unknown in $\mathbf{x}_n$, i.e., in the previous example, $\mathbf{m}_4 = [1, 0, 1, 0]^T$. The missing data indicator vector, $\mathbf{m}_n$, is also known as response indicator vector. $\mathbf{X}$ is the input data set, with $[d \times N]$ dimensions, $\mathbf{T}$ is target set, with $[1 \times N]$ dimensions, and $\mathbf{M}$ is a binary matrix, with $[d \times N]$ dimensions, which is generally known as missing data indicator matrix. According to $\mathbf{M}$, $\mathbf{X}$ is divided into two parts:

$$\mathbf{X} = \{\mathbf{X}_\mathrm{o}, \mathbf{X}_\mathrm{m}\} \tag{2}$$

where $\mathbf{X}_\mathrm{o}$ and $\mathbf{X}_\mathrm{m}$ are, respectively, the observed input set (complete patterns) and the unknown input set (incomplete patterns).

At this point, a simple example is used for clarifying the missing data problem in classification tasks. Suppose a classification problem of N bidimensional input patterns $\mathbf{x} = [x_1, x_2]$, where each pattern can belong to one of two possible classes labelled with 1 or 0. Moreover, any attribute of any input vector can be a missing value. Figure 3 represents complete and incomplete patterns using scatter diagrams.

Figure 3a represents a complete data set; meanwhile, Fig. 3b shows an incomplete version of the data set, where the 33% attribute values are unknown. We represent an incomplete pattern $\mathbf{x}_n = [x_{1n}, ?]$ as a vertical line with horizontal component x_{1n}; and $\mathbf{x}_n = [?, x_{2n}]$ as a horizontal line with vertical component x_{2n}. In higher dimensions, an

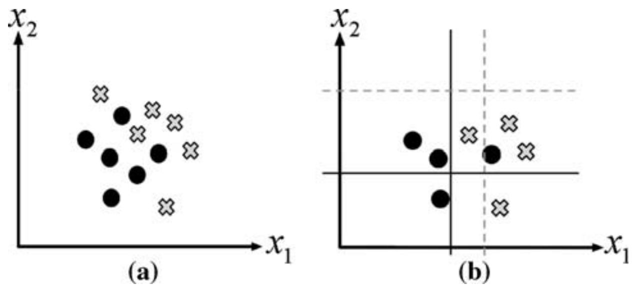


Fig. 3 Two-class classification problems with bidimensional patterns. When missing values occur, they are represented by a *vertical* or *horizontal line*, depending on which attribute is unknown

incomplete pattern could be represented with a line, plane, etc., depending on the number of missing values.

3 Missing data mechanisms

As we have already mentioned in the introduction, missing data is a common drawback that appears in many real-world situations. In surveys [9, 10], it is virtually assured that a certain level of non-response will occur, e.g., partial non-response in a poll or entered data is partially erroneous. In control-based applications, such as traffic monitoring [11], industrial processes [12], or management of telecommunications and computer networks [13], missing values appear due to failures of monitoring or data collector equipment, disruption of communication between data collectors and the central management system, failure during the archiving system (hardware or software), etc. Wireless sensor networks also suffer incomplete data due to different reasons, such as power outage at the sensor node, random occurrences of local interferences or a higher bit error rate of the wireless radio transmissions [14, 15]. In automatic speech recognition, speech samples that are corrupted by very high levels of noise are also treated as missing data [16, 17]. Another example is incomplete observations in financial and business applications like credit assignment (unknown information about the credit petitioner) or financial-time series forecasting (there is no stock price data available for regular holidays) [18, 19]. A recent application in which incomplete data appear is biology research with DNA microarrays, where the gene data may be missing due to various reasons such as scratch on the slide or contaminated samples [20, 21]. Missing values are common in medical diagnosis, e.g., a medical practitioner may not order a test whose outcome appears certain or not relevant to the diagnosis, or a feature can be missing because it proved to be difficult/harmful to measure [22–25].

The appropriate way to handle incomplete input data depends in most cases on how data attributes became missing. The missing data mechanism is characterized by the conditional distribution of **M** given **X**,

$$p(\mathbf{M}|\mathbf{X},\xi) = p(\mathbf{M}|\mathbf{X}_o,\mathbf{X}_m,\xi), \tag{3}$$

where ξ denotes the unknown parameters which defines the missing data mechanism. Little and Rubin [6] define several unique types of missing data mechanisms.

- *Missing completely at random* (*MCAR*). An MCAR situation occurs when the probability that a variable is missing is independent of the variable itself and any other external influences. The MCAR condition can be expressed by the relation

  $$p(\mathbf{M}|\mathbf{X}_o,\mathbf{X}_m,\xi) = p[\mathbf{M}|\xi], \tag{4}$$

  which means that the missingness does not depend on the input values. The available variables contain all the information to make the inferences. Typical examples of MCAR are a tube containing a blood sample of a study subject that is broken by accident (so the blood parameters cannot be measured) or a questionnaire of a study subject that is accidentally lost. The reason for missingness is completely at random, i.e., the probability that an observation is missing is not related to any other patient characteristics.
- *Missing at random* (*MAR*). The missingness is independent of the missing variables but the pattern of data missingness is traceable or predictable from other variables in the database. The MAR condition can be expressed by the relation

  $$p(\mathbf{M}|\mathbf{X}_o,\mathbf{X}_m,\xi) = p[\mathbf{M}|\mathbf{X}_o,\xi], \tag{5}$$

  where the missingness depends only on the observed input data. An example is a sensor that fails occasionally during the data acquisition process due to power outage. In this example, the actual variables where data are missing are not the cause of the incomplete data. Instead, the cause of the missing data is some other external influence.
- *Not missing at random* (*NMAR*). The pattern of data missingness is non-random and depends on the missing variable. In contrast to the MAR situation, the missing variable in the NMAR case cannot be predicted only from the available variables in the database. If a sensor cannot acquire information outside a certain range, its data are missing due to NMAR factors. Then the data are said to be censored. If missing data are NMAR, valuable information is lost from the data; and there is a no general method of handling missing data properly.

When data are MCAR or MAR, the missing data mechanism is termed ignorable. Ignorable mechanisms are important, because when they occur, a researcher can ignore the reasons for missing data in the analysis of the data, and thus simplify the methods used for missing data analysis [7]. For this reason, the majority of research covers the cases where missing data are of the MAR or the MCAR type. In the next sections, several approaches to deal with missing values are explained.

For the focus of this work, we can also divide the missingness mechanisms according to the target to be predicted:

- *Informative*. The fact that a value is missing provides information about a classification target.
- *Non-informative*. The distribution of missing values is the same for all the classes of values.

An example for informatively missing values would be a customer database, where fewer values are missing for active customers than for passive ones, because the company knows its active customers better. These categories of missingness mechanisms make a difference for prediction, because informatively missing values should be treated as extra values for the respective feature (thus arriving at a complete data set), as this information provides hints about the target, which can be learned. For non-informatively missing values, missing value treatment methods, such as imputation, may be used instead, where an algorithm can no longer identify which values are actually missing. Also, most missing value handling methods embedded in learning algorithms are only appropriate for the case of non-informatively missing values.

When discussing the application of machine learning algorithms to data sets including missing values, two scenarios can be distinguished:

- It may be that the data used for training are complete, and values are missing only in the test data.
- Values may be missing both in the training and test data.

The first case is not a common situation, because the classifier has been designed using complete cases, and no assumption about missing data has been done during its training. In this situation, there are two possible solutions: incomplete cases have to be excluded from the data set, or missing values have been filled in by estimations (imputed). This preprocessing is often necessary to enable the training of models, e.g., when using standard multi-layer perceptrons (MLPs), which cannot deal with missing values. The second itemized case is more natural, when the training and test data are gathered and processed in a highly similar way, and so, the classifier is trained considering that input vectors may be incomplete.

4 Complete and available data analysis

When faced with missing values, the complete case analysis and available case analysis are common, but not recommended, procedures used in order to force the incomplete data set into a rectangular complete-data format [6, 7]. The first approach is also known as listwise or casewise deletion, i.e., omit the cases that have missing data for any variable used in a particular analysis. This procedure can be justified whenever large quantities of data are available (in general, only 5% of missing data is an acceptable amount to be deleted from the data set). Its advantage is the possibility of directly using the standard pattern classification methods for complete data. However, a test pattern with missing values cannot be classified because the deletion process will omit it. In available-case analysis only the cases with available variables of interest are used. This method is also known as pairwise deletion, e.g., if the covariance between two variables has to be estimated, only the examples presenting these two variables will be used. In general, available case analysis makes better use of the data than complete case analysis (this method uses all the available values); but its disadvantage is that the number of attributes is not the same for all patterns (it depends on which features are incomplete for each pattern), and so the standard classification methods cannot be directly applied. These approaches are simple, but the major drawback of both procedures is the loss of information that occurs in the deletion process.

5 Imputation methods based on statistical analysis

In this section, we discuss statistical methods that impute (that is, estimate and fill-in) the attribute values of patterns that are missing [6–8]. These methods can be applied to impute one value for each missing item (single imputation) or, in some cases, to impute more than one value, in order to allow appropriate assessment of imputation uncertainty (multiple imputation). In particular, some of the most popular statistical methods for handling missing data are described below.

5.1 Mean imputation

The earliest used method of imputation was unconditional mean imputation. In this approach, missing components of a vector are filled in by the average value of that component in all the observed cases. Consider that there are missing values in the i-th attribute; then this approach performs the imputation stage by computing the mean estimator,

$$\tilde{x}_i = \frac{1}{N_{\mathrm{obs},i}} \sum_{n=1}^{N} (1 - m_{in}) x_{in} \tag{6}$$

where $N_{\mathrm{obs},i}$ is the number of observed values in x_i. Another possibility is class-conditional mean imputation, where missing data of an input pattern are estimated by the average from the complete cases that belongs to the same class as the incomplete pattern. This method has the obvious disadvantages that it under represents the variability in the data, and it also completely ignores the correlations between the various components of the data [7].

5.2 Regression imputation

Regression imputation is well suited when the missing variables of interest are correlated with the data that are available in the complete sample. The missing components are filled in by the predicted values from a regression analysis using the components of the vector that are present. Consider that i-th input feature contains missing values, and the remaining $d-1$ attributes are complete. In this procedure, a regression model $f(\cdot)$ is trained to approximate the unknown feature using the available data,

$$\tilde{x}_i = f(\mathbf{x}_{\text{obs}}) \approx x_i. \tag{7}$$

where $\mathbf{x}_{\text{obs}}$ is the input vector composed of the $d-1$ complete attributes. The complete set ($\mathbf{X}_{\text{o}}$) is used for computing the unknown parameter which defines $f(\cdot)$. If there are two or more incomplete features, a multivariate regression model has to be implemented in order to perform the imputation stage. The method of regression to be used depends on the nature of the data [6]. Linear regression can be used if the variable dependency follows a linear relationship. On the other hand, in non-linear regression the purpose is to fit a curve to the data and to find the required points from the curve. An advantage of this approach over mean substitution is that it preserves the variance and covariance of variables with missing data. When regression is used to impute missing values on independent variables, this will contribute to multi-colinearity, because the imputed values for missing data will be perfectly correlated with the rest of the variables in the model. The disadvantage of this approach is that all the imputed values follow a single regression curve and cannot represent any inherent variation in the data [6, 7].

5.3 Hot and cold deck imputation

Hot deck imputation replaces the missing data with the values from a similar complete data vector, i.e., the data vector that is closest in terms of the components that are present in both vectors for each case with a missing value [7]. The missing components of the incomplete data vector are then substituted with the most similar corresponding components of the matching complete vector. Hot deck imputation has the shortcoming that the estimate of the missing data components are based on a single complete vector in the data set, ignoring any global properties of the data set. Another possibility is the cold deck imputation method, which is similar to the hot deck method, but in it the data source must be different from the current data set [6].

5.4 Multiple imputation

The three described approaches above provide a simple missing data imputation, which does not reflect the uncertainty about the prediction of the unknown values. Instead of filling in a single value for each missing one, a multiple imputation procedure replaces each missing value with a set of plausible ones that represent the uncertainty about the right value to impute [9]. The missing values are imputed M times to produce M complete data sets using an appropriate model that incorporates random variation. This method, which is shown in Fig. 4, is used for general purpose handling of missing data in multivariate analysis.

The desired analysis is performed on each data set using the standard complete data methods, and then the average of parameter estimates across M samples is taken to produce a single point estimate. Standard errors are calculated as a function of average squared standard errors of the M estimates and the variance of the M parameter estimates across samples [8]. Thus, multiple imputation procedure involves three distinct phases:

- The missing data is filled in M times to generate M complete data sets.
- The M complete data sets are analyzed by using standard procedures.
- The results from the M complete data sets are combined for the inference.

Little and Rubin [6] conclude that casewise and mean substitution methods are inferior when compared with multiple imputation. Regression methods are somewhat

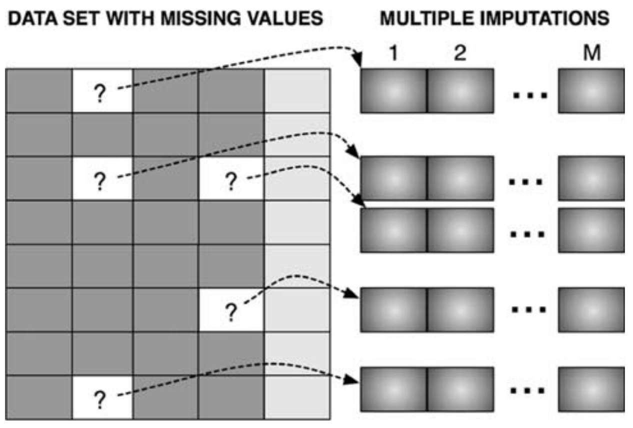


Fig. 4 Schematic representations of multiple imputations. The missing values are imputed M times to produce M complete data sets using an appropriate model that incorporates random variation

better, but not as good as multiple imputation or other maximum-likelihood approaches.

6 Imputation methods based on machine learning

Imputation methods based on machine learning are sophisticated procedures, which generally consist of creating a predictive model to estimate values that will substitute those missing. These approaches model the missing data estimation based on information available in the dataset. There are several options varying from imputation with K-nearest neighbor (K-nn) to imputation procedures based on auto-associative neural networks (AANN). Most of them are focused on imputing missing data, and so, after imputed data fill in the incomplete feature values, it is necessary to train a classifier using the imputed training set.

6.1 Imputation with *K*-nn

The K-nearest neighbor (K-nn) method is a common hot deck method, in which K nearest neighbors (donors) are selected from the complete cases, so that they minimize a similarity measure. The nearest, most similar, neighbors are found by minimizing a distance function [26]. Given an incomplete pattern $\mathbf{x}$,

$$V = \{v_k\}_{k=1}^{K}. \tag{8}$$

represents the set of its K nearest neighbors (according to a distance metric) arranged in increasing order of its distance. Once the donors have been found, a replacement value to substitute the missing attribute value must be estimated. How the replacement value is calculated depends on the type of data; the mode can be used for discrete data and the mean for continuous data. An improved alternative is to weight the contribution of each of the K neighbors according to their distance to the incomplete pattern whose values will be imputed, giving greater contribution to close donors. Rather than using only the complete instances in $\mathbf{X}$, i.e., $\mathbf{X}_o$, a better approach is to select the K closest cases from the training patterns with known values in the attributes to be imputed. One of the key aspects of the K-nn imputation approach is the distance measure. A well-known and accurate distance function is the heterogeneous euclidean overlap metric (HEOM) [26, 27]. Consider a pair of input vectors, represented by $\mathbf{x}_a$ *y* $\mathbf{x}_b$, the HEOM distance between them is:

$$D(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{\sum_{i=1}^{n} D_i(x_{ia}, x_{ib})^2}. \tag{9}$$

where $d_i(x_{ia}, x_{ib})$ is the distance between $\mathbf{x}_a$ and $\mathbf{x}_b$ on its i-th attribute:

$$D_i(x_{ia}, x_{ib}) = \begin{cases} 1, & (1 - m_{ia})(1 - m_{ib}) = 0, \\ D_0(x_{ia}, x_{ib}), & x_i \text{ is a discrete attribute,} \\ D_N(x_{ia}, x_{ib}), & x_i \text{ is a numerical attribute.} \end{cases} \tag{10}$$

Unknown data are handled by returning a distance value of 1 (i.e., maximal distance) if either of the input values is unknown. The overlap distance function D_0 assigns a value of 0 if the discrete attributes are the same; otherwise, a distance value of 1. The range normalized difference distance function D_N is

$$D_N(x_{ia}, x_{ib}) = \frac{|x_{ia} - x_{ib}|}{\max(x_i) - \min(x_i)}. \tag{11}$$

where $\max(x_i)$ and $\min(x_i)$ are, respectively, the maximum and minimum values observed in the training set for the numerical attribute x_i.

Batista and Monard compare K-nn with the machine learning algorithms C4.5 and C2 and conclude that K-nn outperforms the other two, and that it is suitable also when the amount of missing data is large [27]. In the context of DNA research, Troyanskaya et al. report on a comparison of three imputation methods [20]. They conclude that the K-nn method is far better than the other tested methods (mean imputation and imputation based on singular value decomposition), and also that it is robust with respect to the amount and type of missing data. An advantage over mean imputation and simple hot deck method (in fact, K-nn with $K = 1$) is that replacement values are influenced only by the most similar cases rather than by all the cases or the most similar one, respectively. The main drawback is that the K-nn method looks for the most similar cases, through all the data set (in the complete data portion), which implies a high computational cost.

6.2 SOM imputation

The self-organizing map (SOM) was originally developed to imitate the formation of the orientation of specific neural cells in the brain [28]. A SOM describes a mapping from a higher dimensional (dimension d) input data space to a lower dimensional (dimension d_L) map space, being $d_\mathrm{L} = 2$ the most extended approach. The basic SOM consists of nodes placed in a d_L-dimensional array. Each node has a d-dimensional weight vector associated to it. Training of the SOM assigns weights to each node that is representative of the input data, and such that nodes that are spatially close in the array have similar weight vectors [28]. For each training input vector $\mathbf{x}$, the neuron with weight vector most similar to $\mathbf{x}$ is called the best matching unit (BMU) or image node. The weights of the BMU and its neighbor nodes close to $\mathbf{x}$ in the SOM lattice are adjusted towards the input vector. A neighborhood function has to be defined, being a Gaussian function a common choice. Generally, a SOM can be considered a non-linear version of PCA.

Samad and Harp implement SOM approaches to handle missing values changing the treatment of the input data [29]. In particular, when an observation with missing features is given as input to the map, the missing variables are simply ignored when distances between observation and nodes are computed. This principle is also applied both for selecting the image-node and for updating weights. The authors also noticed that performances are always better when incomplete examples are used during the weight update, compared to training on the complete examples only. Fessant and Midenet extend this procedure to missing data imputation [30]. The SOM-based imputation model is illustrated in Fig. 5.

First, when an incomplete pattern is presented to the SOM, its image-node is chosen ignoring the distances in the missing variables; second, an activation group composed of image-node's neighbors is selected; and finally, each imputed value is computed based on the weights of the activation group of nodes in the missing dimensions. In [30], this approach is compared with hot-deck and standard multi-layer perceptron (MLP) based imputation, and it is concluded that SOM works better than the other two methods, emphasizing that SOM-based method requires less learning observations than other models, like MLP; incomplete observations can also be used during the training stage. Following this idea, Piela implements missing data imputation in a tree structured self-organizing map (TS-SOM) [31], which is made of several SOMs arranged to a tree structure. The major advantages of this approach over the basic SOM are its faster convergence and its computational benefit when the number of input vector is large.

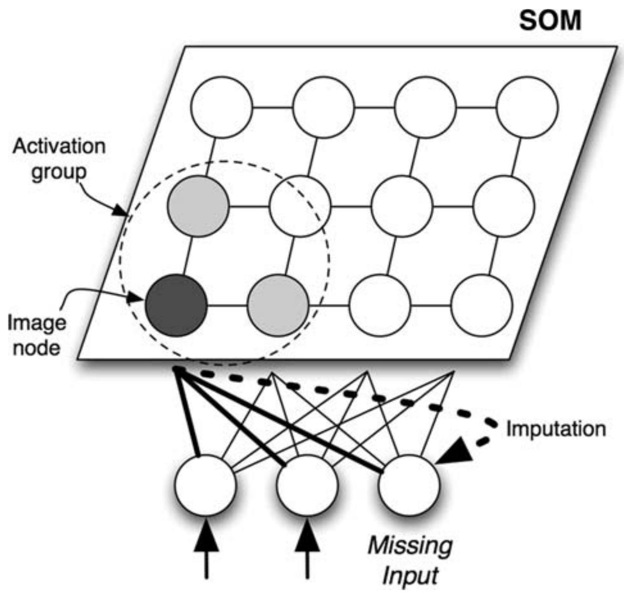


Fig. 5 Self-organizing map model for imputation. For each input vector with missing values, its image-node is chosen only measuring the distances with the known attributes; after that, each missing value is imputed based on the weights of the activation group of nodes in the incomplete attributes

6.3 Multi-layer perceptron imputation

A basic multi-layer perceptron (MLP) imputation approach consists of training an MLP using only the complete cases as regression model: given d input features, each incomplete attribute is learned (it is used as output) by means of the remaining complete attributes given as inputs. The MLP imputation scheme can be described as follows:

1. Given an incomplete input dataset $\mathbf{X}$, separate the input vectors that do not contain any missing data (observed component, $\mathbf{X}_o$) with the ones that have missing values (missing component, $\mathbf{X}_m$).
2. For each possible combination of incomplete attributes in $\mathbf{X}_m$, construct an MLP scheme using $\mathbf{X}_o$. The target variables are the attributes with missing data, and the input variables are the other remaining attributes [32]. In this approach, there is one MLP model per missing variables combination. Depending on the nature of the attributes to be imputed (continuous or discrete), different error functions (sum of squares error or cross-entropy error) are minimized during the training process.

After the optimal MLP architectures are chosen, for each incomplete pattern in $\mathbf{X}_m$, unknown values are predicted using its corresponding MLP model (according to the attributes to be imputed). Figure 6a illustrates this procedure with four attributes, of which the third one has some unknown values.

Sharpe and Solly use this MLP scheme on a medical diagnosis application [32]. Nordbotten also experiments this approach in survey data [33]. Gupta and Lam use it for predicting the increase or decrease of the earnings of a company using some financial variables as inputs [34]. The MLP approach is a useful tool for reconstructing missing values. However, its main disadvantage is that when missing items appear in several attributes, several MLP models have to be designed, one per missing variables combination. Other MLP methods have been proposed to impute missing values using a different solution [35, 36]. In [35], Yoon and Lee propose the TEST (Training-EStimation-Training) algorithm as a way of using MLP to impute missing data. It consists of three steps. First, the network is trained with all the complete patterns. Second, the parameters (weights) are used to estimate the missing data in all incomplete patterns by means of back-propagation in the inputs. Third, the MLP network is trained

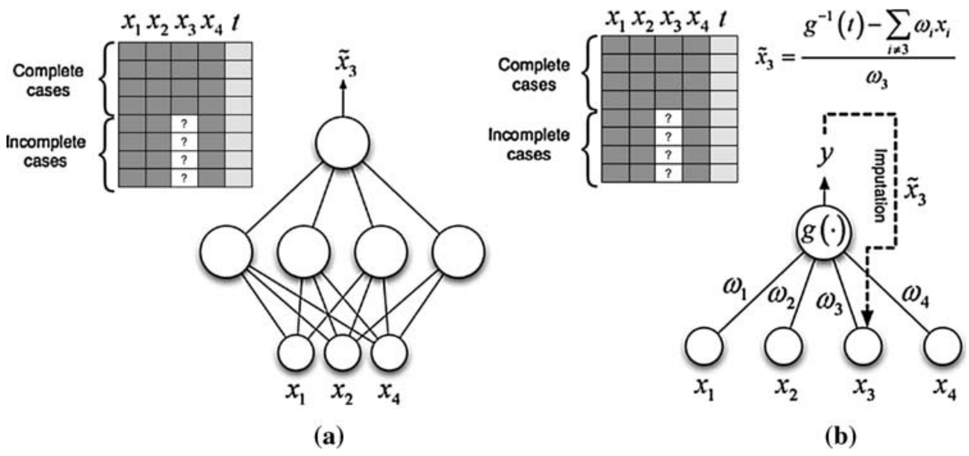


Fig. 6 Two different ANN solutions for a classification problem of input vectors with four attributes, where the attribute x_3 is incomplete. First, in (**a**), the standard imputation using an MLP as regressor for the incomplete attribute x_3 is shown. Second, in (**b**), how a single perceptron can estimate missing values using its inverse output is illustrated

again using the whole data set, i.e., complete and imputed patterns. However, this procedure cannot estimate missing values in the test set. Following this idea, Kallin [36] implements an imputation procedure in a single layer perceptron (SLP). In a first step, the SLP is trained using only the complete cases; after that, imputation is done with the inverse of the obtained equation for the desired classification output (it requires that the pre-processing and activation functions can be inverted). This approach is shown in Fig. 6b. According to Ref. [36], it is a simple solution, which produces good results that are comparable to multiple imputation.

6.4 Recurrent neural network imputation

A recurrent neural network (RNN) is an architecture with feedback connections from its units. Bengio and Gingras propose RNN with feedback into the input units for estimating missing data [37]. First, the missing values are initialized with the mean imputation, and these values are updated using the feedback connections, while the network is trained to learn the classification task. The missing values are modified as a function of the missing input in the last iteration and the weighted sum of a set of recurrent links from the other units (hidden and missing) to the missing unit with a unit delay. Figure 7 illustrates the RNN approach used in [37]. The authors show that when there are dependencies among input variables, the output prediction can be improved by taking them into account; the recurrent network performs significantly better than a standard network with missing values replaced by their mean. Some works have extended the RNN using true values of incomplete features as extra targets during the training [38].

6.5 Auto-associative neural network imputation

An auto-associative neural network (AANN) is a set of neurons that are completely connected in the sense that

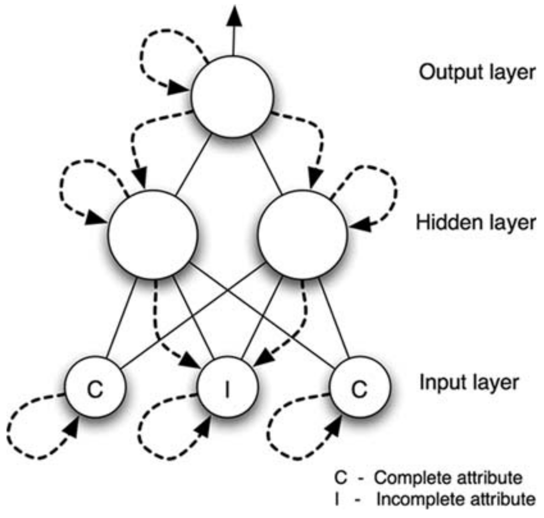


Fig. 7 Missing data imputation with recurrent neural networks (RNN). Missing values are imputed using the feedback connections from the hidden neurons

each neuron receives input from, and sends output to, all the other neurons. Some works tackle the missing data imputation by means of this kind of networks [39–43]. Figure 8 shows an AANN, which is trained to estimate missing data. First, the network learns from complete cases, in order to replicate all of the inputs as outputs. Second, when unknown values are detected, the weights are not updated. Instead, the missing values are replaced by the network outputs.

6.6 Multi-task learning approaches

In recent years, some approaches have been proposed using the advantages of multi-task learning (MTL) [44–47]. MTL is an approach to machine learning that solves a problem (main task) together with other related problems (extra tasks) at the same time, using a shared representation [44–46]. In [47], an MTL neural network scheme that

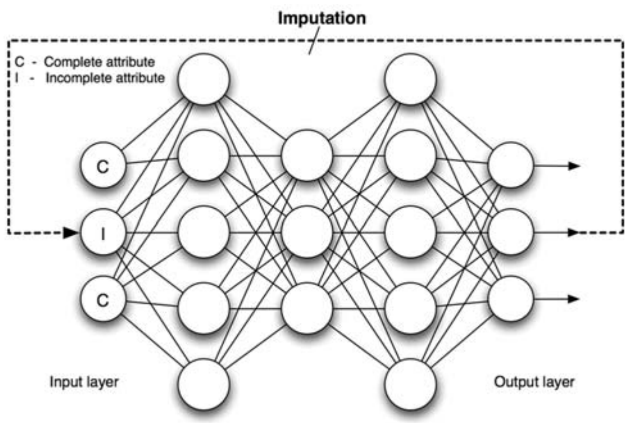


Fig. 8 Missing data imputation with auto-associative neural networks (AANN). Missing data imputation is done using the output unit which learns the corresponding incomplete attribute

combines missing data imputation and pattern classification is proposed. This procedure utilizes the incomplete features as secondary tasks, and learns them in parallel with the main classification task. Figure 9 shows a full-connected neural network based on MTL for solving a decision problem what presents missing values in the features x_2 and x_3. This neural scheme learns three different tasks: one classification task, associated with the network output y, and two imputation tasks, associated with the outputs $\tilde{x}_2$ and $\tilde{x}_3$.

In the input layer, there is an input unit for each feature and also an extra input unit associated to the classification target. This extra input (classification target) is used as hint to learn the secondary imputation tasks. Hidden neurons used in this approach do not work in the same way as standard neurons, because they compute different outputs for the different tasks to be learned [47]. They do not include the input signal they have to learn in its corresponding output unit in the sum product. For example, in Fig. 9, the imputation output $\tilde{x}_2$ is learned using the information from the attributes x_1, x_3, and x_4, and the classification target t, but it does not depend on x_2. The imputation outputs are used to estimate the missing values during the training stage. By doing this, missing data estimation is oriented to solve the classification task, because the learning of the classification task affects the learning of the secondary imputation tasks. According to [47], this approach outperforms other well-known techniques, such as GMM trained with EM and K-nn, but its major drawback is that it uses the quadratic error as cost function to be minimized during the training, which does not consider the input data distribution.

6.7 Performance metrics for missing data imputation

The performance and capabilities of the different imputation approaches in classification problems can be evaluated considering the two kinds of tasks to be solved, i.e.,

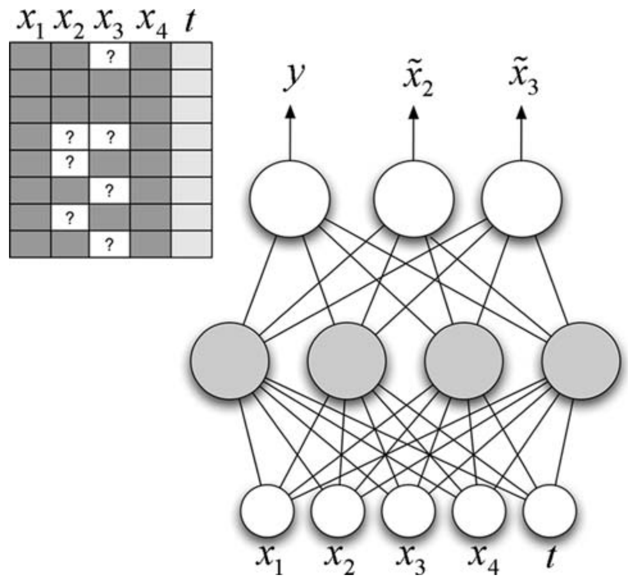


Fig. 9 MTL neural network for solving a classification problem of input vectors with four attributes, where the features x_2 and x_3 are present incomplete data. Hidden neurons used in this approach do not work in the same way as standard neurons, because they compute different outputs for the different tasks to be learned. This network learns the classification task, and each imputation task associated to an incomplete input feature at the same time. By doing this, missing data imputation is oriented to solve the main classification task

classification and imputation tasks. In the first case, once the missing values have been imputed, a classifier is trained, and its accuracy is measured computing the classification error rate (CER) over the test patterns [1–3], whereas, for measuring the quality of the missing data estimation, it is needed to insert incomplete data in an artificial way for different missing data percentages and different combination of attributes. Let us consider that $\tilde{x}_i$ denotes the imputed version of the i-th attribute, and $\hat{x}_i$ denotes the true version of the same variable. Two different criteria can be used for comparing the imputation methods:

- *Predictive accuracy* (*PAC*). An imputation method should preserve the true values as far as possible. Considering that the i-th attribute has missing values in some input patterns, its imputed version $\tilde{x}_i$ must be close to the $\hat{x}_i$ (variable with true values). The Pearson correlation between $\tilde{x}_i$ and $\hat{x}_i$ provides a good measure of the imputation performance, and its it given by

$$\mathrm{PAC} \equiv r = \frac{\sum_{n=1}^{N}\left(\tilde{x}_{i,n} - \bar{\tilde{x}}_i\right)\left(\hat{x}_{i,n} - \bar{\hat{x}}_i\right)}{\sqrt{\sum_{n=1}^{N}\left(\tilde{x}_{i,n} - \bar{\tilde{x}}_i\right)^2\left(\hat{x}_{i,n} - \bar{\hat{x}}_i\right)^2}} \tag{12}$$

where $\tilde{x}_{i,n}$ and $\hat{x}_{i,n}$ denote, respectively, the n-th value of $\tilde{x}_i$ and $\hat{x}_i$, and besides, $\bar{\tilde{x}}_i$ and $\bar{\hat{x}}_i$ denote, respectively, the mean of the N values included in $\tilde{x}_i$ and $\hat{x}_i$. A good imputation method will have a value for the Pearson correlation close to 1.

- *Distributional accuracy* (*DAC*). An imputation method should preserve the distribution of the true values. One measure of the preservation of the distribution of the true values is the distance between the empirical distribution function for both the imputed and the true values. The empirical distribution functions, $F_{\hat{x}_i}$ for the cases with true values, and $F_{\tilde{x}_i}$ for the cases with imputed values, are defined as

$$F_{\hat{x}_i}(x) = \frac{1}{N}\sum_{n=1}^{N} I(\hat{x}_{i,n} \leq x) \tag{13}$$

$$F_{\tilde{x}_i}(x) = \frac{1}{N}\sum_{n=1}^{N} I(\tilde{x}_{i,n} \leq x) \tag{14}$$

where I is the indicator function. The distance between these functions can be measured using the Kolmogorov–Smirnov distance, D_{KS}, which is given by

$$\mathrm{DAC} \equiv D_{\mathrm{KS}} = \max_{n}(\|F_{\hat{x}_i}(x_n) - F_{\tilde{x}_i}(x_n)\|) \tag{15}$$

where the x_n values are the jointly ordered true and imputed values of attribute x_i. A good imputation method will have a small distance value.

In many scenarios, the CER is the most significant metric due to the fact that the main objective is to solve a classification problem, and the imputation is a secondary task whose aim is to provide imputed values which help to solve the classification problem. For example, given two or more imputation methods and a neural network classifier, the best imputation approach will be that method which provides better CER in the test set given the same weight initialization for the classifier.

7 Model-based procedures

The methods described in this section are called model-based methods since the researcher must make assumptions about the joint distribution of all the variables in the model. One of the most used approaches in this category is the mixture models trained with the expectation–maximization (EM) algorithm. Consider that the input vectors are generated following a probability density function (pdf) $p(\mathbf{x}|\boldsymbol{\theta})$, being $\boldsymbol{\theta}$ the parameters which defines this pdf. Given N input vectors,

$$p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\theta}). \tag{16}$$

is the likelihood of $\boldsymbol{\theta}$ given $\mathbf{X}$. In maximum likelihood (ML) estimation, we wish to estimate the model parameter(s) for which the observed data are the most likely. In addition to this, mixture models provide a general semi-parametric model for arbitrary densities [1–3], where the density functions are modeled as a linear combination of J component densities (in a non-parametric kernel-based approach, it is a linear combination of kernel or basis functions). Thus, this approach considers that $p(\mathbf{x})$ can be approximated by

$$p(\mathbf{x}|\boldsymbol{\theta}) \approx \sum_{j=1}^{J} p\left(\mathbf{x}|\theta_j, j\right) P(j). \tag{17}$$

where θ_j are the parameters of the j-th component and $P(j)$ are the mixing parameters [1–3]. In particular, real valued data can be modeled as a mixture of Gaussians; and for discrete valued data, it can be modeled as a mixture of Bernoulli densities or as a mixture of multinomial densities [48, 49]. From Eqs. 16, 17 and the independence assumption, we see that the log-likelihood given the dataset is

$$L(\boldsymbol{\theta}|\mathbf{X}) = \sum_{n=1}^{N} \log \sum_{j=1}^{J} p\left(\mathbf{x}_n|\theta_j, j\right) P(j). \tag{18}$$

Following a ML estimation, the best model has parameters that maximize $L(\boldsymbol{\theta}|\mathbf{X})$. However, this function is not easily maximized numerically because it involves the log of a sum. Intuitively, there is a 'credit assignment' problem: it is not clear which component of the mixture generated a given input data vector and thus which parameters to adjust for fitting this input vector. The EM algorithm is an efficient iterative procedure for solving this credit assignment problem. This method computes the ML estimate in the presence of missing or hidden data. The intuition is that if one had access to a 'hidden' random variable $\mathbf{z}$ that indicated which instance is generated by which mixture component, then the maximization problem would decouple into a set of simple maximizations. Using the hidden variables, a 'complete-data' log likelihood function can be written:

$$L_c(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Z}) = \sum_{n=1}^{N}\sum_{j=1}^{J} z_{nj} \log p\left(\mathbf{x}_n|\mathbf{z}_n, \theta_j\right) P(\mathbf{z}_n, \theta_j) \tag{19}$$

which does not involve a log of summation. Since $\mathbf{z}$ is unknown, L_c cannot be utilized directly, so we instead work with its expectation in a iterative procedure, denoted by $Q(\boldsymbol{\theta}|\boldsymbol{\theta}_s)$, where $\boldsymbol{\theta}_s$ are the model parameters in the s-th iteration. As shown in [50], $L(\boldsymbol{\theta}|\mathbf{X})$ can be maximized by iterating the following two steps:

$$\begin{aligned} &E \text{ step: } Q(\boldsymbol{\theta}|\theta_s) = E[L_c(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Z})|X, \theta_s], \\ &M \text{ step: } \theta_{s+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\theta_s). \end{aligned} \tag{20}$$

The expectation or E-step computes the log likelihood of the data, and the maximization or M-step finds the parameters that maximize this likelihood [50]. The EM algorithm has been successfully applied to a variety of problems involving incomplete data, such as training Gaussian mixture models (GMMs) [48, 49]. EM associates a given incomplete-data problem with a simpler complete-data problem, and iteratively finds the maximum likelihood estimates of the missing data. In a typical situation, EM converges monotonically to a fixed point in the state space, usually a local maximum.

Applying EM to an incomplete-data problem can actually be perceived as a special instance of local search. For example, when learning a mixture of Gaussians [48, 49], the state space consists of all the possible assignments to the means, covariance matrices, and prior probabilities of the Gaussian distributions. Missing features can be treated naturally in this framework, by computing the expectation of the sufficient statistics for the E-step over the missing values as well as the mixture components. Thus, in the expectation, or E-step, the missing data are estimated given the observed data and current estimate of the model parameters. This is achieved using the conditional expectation. In the M-step, the likelihood function is maximized under the assumption that the missing data are known. The estimates of the missing data from the E-step are used instead of the actual missing data. The EM algorithm starts with some assignment to these state variables; for example, the mean vectors are usually initialized with K-means, and iteratively update the state variables until it converges to a state with a locally maximum likelihood estimate of training data. Convergence is assured since the algorithm is guaranteed to increase the likelihood at each iteration. In a classification problem, the mixture-modeling framework can model the class label as a multinomial variable, i.e., the mixture model estimates the joint probability that an input vector has determined attributes and belongs to a determined class. Once the model is obtained, the most likely label for a particular input pattern may be obtained by computing the class-posterior probabilities using the Bayes theorem [1–3, 48, 49].

In addition to this, efficient neural network approaches for handling missing data have been developed to incorporate the input data distribution modeling into the network [51–54]. Ahmad and Tresp discuss Bayesian techniques for extracting class probabilities given partial data [51]. Considering that the classifier outputs are good estimates of the class-posterior probabilities given the input vector, and it can be split up into a vector of complete features and other with unknown features, they demonstrate that the optimal solution involves integrating over the missing dimensions weighted by the local probability densities. Moreover, in [51], closed-form approximations to the class probabilities are obtained by using Gaussian basis function networks, and they extend to an MLP classifier calculating the integrals with Monte Carlo techniques. Instead of doing a numerical approximation of these integrals, Tresp et al. [52, 53] propose an efficient solution using GMMs and Parzen windows to estimate the conditional probability densities that appear in the integral (for more details, see Refs. [51–53]). Finally, Williams et al. have recently proposed a classification scheme based on logistic regression with missing values [54]. In this approach, the conditional density functions are estimated with a GMM whose parameters are obtained using both EM and variational Bayesian EM (VB-EM). According to Ref. [54], this method outperforms other procedures, such as GMM trained with EM, mean and multiple imputation, but its major drawback is the restriction to linear classifier. Finally, we would like to remark an efficient procedure based on Bayesian networks, which has been proposed by Ramoni and Sebastani [55]. This procedure is known Robust Bayesian estimator (RBE), and it learns conditional probability distributions from incomplete data sets without making any assumption on the missing data mechanism. Its main advantage is the robustness with respect to the different types of missing data. This is achieved by considering probability intervals containing all possible estimated values, instead of a single value, that can be learned from all completed data sets. The width of these intervals monotically increases function of the information available in the dataset, and thus provides a measure conveyed by the data.

8 Some machine learning approaches for handling missing data

Up to now, this work has discussed several machine learning procedures where missing values are replaced by a plausible value, i.e., missing values are imputed with an estimator, and afterwards that classification is done with another machine. Conversely, other approaches have been proposed for handling missing data in classification problems avoiding explicit imputations. Now, we will summarize the most representative machine learning procedures which are able to deal with unknown values, such as ensemble methods, decision trees, fuzzy approaches and support vector solutions.

8.1 Neural network ensembles

Neural network ensemble models have also been used for classification of incomplete data [32, 56–58]. Sharpe and Solly propose a procedure named network reduction [32]. In this method, a set of MLPs is created, and each MLP classifies basing on each different possible combination of complete features, in order to cover the complete range of attributes with missing values. The main drawback of this method is that it requires a huge number of neurons when multiple combinations of incomplete attributes are presented. Krause and Polikar [56] develop an ensemble of classifiers trained with random subsets of features for missing feature problem. As each classifier component of the ensemble is generated according to a weighted vector for features, it cannot guarantee that all the instances have been trained on the ensemble. Therefore, useful information maybe ignored. In Ref. [57], the incomplete dataset is divided into a group of complete sub data sets, which is then used as the training sets for the neural networks. According to Ref. [57], compared with other methods dealing with missing data in classification, this method can utilize all the information provided by the data with missing values, maintaining maximum consistency of incomplete data and avoiding the dependency on distribution assumption. Juszczak and Duin propose to form an ensemble of one-class classifiers trained on each feature [58]. Thus, when any feature values are missing for a data point to be labeled, the ensemble can still make a reasonable decision based on the remaining classifiers.

8.2 Decision trees

In this kind of methods, we stand out three well-known approaches: ID3, C4.5, and CN2. These procedures can handle missing values in any attribute for both training and test sets [59–64]. ID3 is a basic top–down decision tree algorithm that handles an unknown attribute by generating an additional edge for the unknown. Thus, an unknown edge has been taken as a new possible value for each attribute and it has been treated in the same way as other values. C4.5 is an extension of ID3 proposed by Quinlan [60]. It uses a probabilistic approach to handle missing values in the training and test data set. In this approach, the way of dealing with missing values changes during the training and testing stages. During training, each value for an attribute is assigned a weight [60–63]. If an attribute value is known, then the weight is equal to one; otherwise, the weight of any other value for that attribute is the relative frequency of that attribute. On the testing phase, if a test case is incomplete, it explores all the available branches (below the current node) and decides the class label by the most probabilistic value. The CN2 is an algorithm for inducing propositional classification rules. It uses a rather simple imputation method to treat missing data. Every missing value is filled in with its attribute most common known value, before computing the entropy measure [64].

8.3 Fuzzy approaches

Many fuzzy procedures have been developed in order to handle missing data [65–77]. Ishibuchi et al. propose an MLP classification system where unknown values are represented by interval inputs [65, 66]. For example, if the pattern space of a particular classification problem is the d-dimensional unit cube $[0, 1]^d$, each unknown feature value is represented by the interval input that includes all the possible values of that attribute. When the attribute value is completely known, it is also represented by intervals (e.g., 0.3 is represented by [0.3, 0.3]). This network is trained by means of back-propagation algorithm for fuzzy input vectors [67]. Gabrys develops a general fuzzy min–max (GFMM) neural network using hyperbox fuzzy sets [68, 69]. A hyperbox defines a region of the d-dimensional pattern space, by its min-point and its max-point, so the patterns contained within the hyperbox have “full class” membership. Learning in the GFMM neural network for classification consists of creating and adjusting hyperboxes in pattern space [69]. This procedure handles missing values in a similar way as the interval inputs, i.e., the missing attribute is modeled by a real valued interval spanning the whole range of values. In Refs. [71–73], some techniques to tolerate missing values based on a system of fuzzy rules for classification are proposed. In general, a fuzzy rule based classifier consists of a set of rules for each possible category, where each rule can be decomposed into individual one-dimensional membership functions corresponding to the fuzzy sets (see Ref. [71] for more details). When an incomplete pattern has to be classified, the rules are obtained using only one-dimensional membership function of the known attributes, which is computationally very efficient. Other fuzzy solutions for missing values are based on fuzzy C-means [74–76].

8.4 Support vector machines

In recent years, some works have extended the standard formulation of support vector machines (SVMs) for handling uncertainty in the input data [78–82]. Bhattacharyya et al. propose a mathematical programming method to deal with uncertainty in the observations of a classification problem [78]. They consider a standard SVM classifier with certainty (all the training data are known), and they extend it to handle missing values replacing the linear classification constraints by a probabilistic one. This is done by modeling the missing variables as random variables; in particular, the input data a drawn from a Gaussian distribution [78]. Moreover, the model parameters (mean and covariance matrices) are estimated by means of the EM algorithm. In [79], Smola et al. shows how SVMs and Gaussian processes (GPs) can deal with missing data. This procedure is based on the fact that kernel methods can be written as estimator in an exponential family. More specifically, GPs can be seen to maximize the negative log-posterior under a normal prior on the natural parameter of the exponential density, whereas SVMs maximize the likelihood ratio [79]. In this approach, estimation with missing values becomes a problem of computing marginal distribution, and finding efficient optimization methods, like the constrained concave convex procedure (CCCP) and the EM algorithm [79]. Pelckmans et al. propose a modified risk function taking into account the uncertainty of predicted outputs when missing values are involved [80]. This is done by incorporating a probabilistic model for the missing data. This method generalizes the approach of mean imputation in the linear case, and the proposed kernel machine reduces to the standard SVM when no input values are missing [80]. Other approaches have been proposed to deal with noisy inputs. Bi and Zhang develop a novel formulation of support vector classification [81], which allows uncertainty in input data, assuming that inputs are subject to an additive noise that follows certain distribution. Recently, Chechik et al. [82] have proposed an accurate classifier for incomplete data using a max-margin learning framework. This method uses a geometrically inspired objective function, solving two optimization approaches: the linearly separable case, which is written as a set of convex feasibility problems, and the non-separable case, that has a non-convex objective which is optimized iteratively. By avoiding a previous imputation stage, this method provides considerable computational savings, and more importantly, its gives an elegant procedure for handling complex patterns of missing values.

9 Experimental evaluation of solutions based on missing data estimation

As we have already mentioned, the main goal of this work is to provide a general overview for incomplete data classification methods, giving a special emphasis to machine learning solutions. A detailed and global experimental evaluation of incomplete data classification methods is beyond of the scope of this work. However, we have selected some of the most representative approaches for missing data imputation in order to show experimental results. Missing data imputation is the most common solution for incomplete data classification in many applications. This pre-processing step is chosen, because the great majority of decision making tools, such as the commonly used artificial neural networks and many other machine learning techniques implemented in commercial software, cannot be used for performing the classification task when data are not complete.

This section evaluates the influence of imputing missing values into the classification accuracy obtained by an artificial neural network. In particular, four missing data estimation techniques have been selected: K-nn imputation, SOM imputation, MLP imputation, and the EM algorithm. In order to compare these approaches, we have chosen a synthetic toy dataset, and two different real classification problems, one without missing values and other with incomplete data. The main reason for using complete datasets is to measure the influence in the results when different percentages of missing values are considered. Missing values are inserted in the same proportion into the training and test sets. In particular, the tested complete problems are an artificial dataset, named Toy problem, and a well-known vowel recognition problem, Telugu.[2] The remaining incomplete dataset is a medical diagnosis problem, Sick-Thyroid.[3] An MLP with one hidden layer is trained to perform the classification task. For each problem, the weights initializations are the same for all the trained classifiers using the edited sets (complete patterns and incomplete patterns with imputed values) obtained by the different imputation approaches. Network weights are computed using the scaled conjugate gradient method during the training stage, and classification results are averages of 20 simulations (i.e., 20 weights initializations).

9.1 Toy problem

This dataset is a two-dimensional synthetic classification problem composed of five Gaussian components, which is modeled on a dataset used by Ripley [2]. Data set is composed by 1,500 patterns where each component has 300 patterns. Training, validation, and test subsets are generated by randomly selecting 100 patterns from each Gaussian component. The input datasets have been normalized (zero mean and unity standard deviation). Missing values (5, 10, 20, 30, 40%) are randomly inserted into the first attribute in training, validation, and test sets. Figure 10 shows the training data with 5% of unknown values. An incomplete pattern is represented by an horizontal line.

9.1.1 First stage: missing data imputation

As the 'true' values are known, we can measure the quality of the missing data estimation using the PAC and DAC measures for the different imputation procedures. However, are they accurate metrics for evaluating the quality of imputations in a classification scenario? In this section, we pretend to show that the selection of the parameter values for an imputation method must be done according to the classification results, and not using PAC or DAC metrics. For doing this, we will make use of one of most common and well-known missing data estimation approaches: K-nn imputation. In this approach, after the distances have been computed using the HEOM metric (9), the number of

[2] http://www.isical.ac.in/∼sushmita/patterns/.

[3] http://archive.ics.uci.edu/ml/.

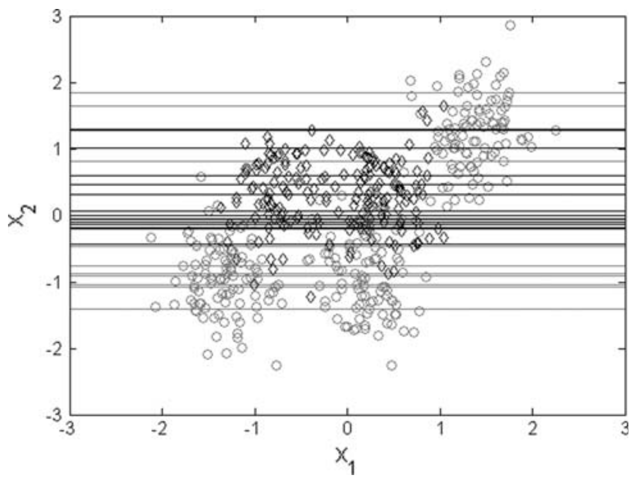


Fig. 10 Toy problem, a synthetic binary classification two-dimensional dataset. Missing values are randomly inserted into the first attribute. An *horizontal line* represents an incomplete pattern

nearest neighbors (K) must be chosen. In particular, the tested values for K are from 1 to 20. Figure 11 shows the obtained PAC and DAC results in the test set for different values of K, considering 5, 10, 20, 30, and 40% of missing data in the first attribute. As it can be observed in this figure, accurate estimations in terms of the PAC metric is obtained with high values of K, and conversely, low values of K provides better imputation results considering the DAC metric.

Figure 12 shows the imputed test sets, with 20% of missing values, using $K = 1$ and $K = 20$. The input data distribution is preserved using the hot-deck approach ($K = 1$), but it can provide high imputation errors according to 'true' values. However, when the imputed values are obtained from a local approximation with $K = 20$, this technique tends to provide the minimum mean square error (MMSE) estimator and its imputation results are accurate (on average) according to the 'true' values, but this solution does not preserve the input data distribution.

Intermediate values for the parameter K can be more satisfactory, and its optimal value must be selected according to the classifier performance, i.e., we have to choose the value of K that provides a better classification accuracy given a decision making tool. Model parameter selection can be typically carried out by choosing a model criterion such as the cross-validation [1–3]. In addition to

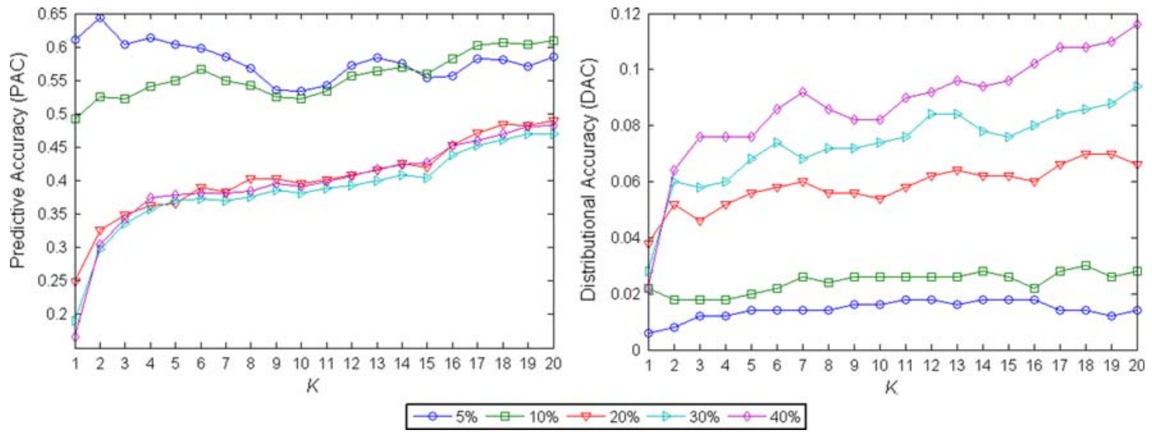
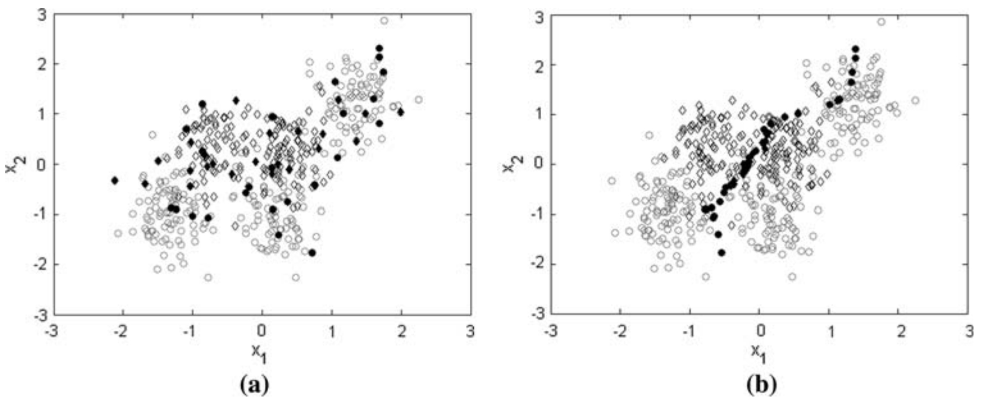


Fig. 11 Predictive accuracy (PAC) and distributional accuracy (DAC) obtained by K-nn imputation using different values for K, when missing values (from 5 to 40%) are artificially inserted in the first attribute of the Toy problem

Fig. 12 Missing data imputation by K-nn using $K = 1$ (**a**) and $K = 20$ (**b**) in Toy problem, considering 20% of missing data in x_1. Input vectors with imputed values are shown in *black color*



this, in real-life incomplete classification problems, the 'true' values for the missing ones are not available, and thus, PAC and DAC metrics cannot be used. All these conclusions are extensible to other missing data estimation approaches, such as MLP, EM or SOM imputation. In particular, the number of hidden neurons for MLP imputation is chosen from 2 to 25, the number of mixture components in EM algorithm from 2 to 20, and the number of nodes in a SOM approach from 10 to 100. For each imputation method, its model parameters are chosen according to the classification results in the validation set.

9.1.2 *Second stage: classification*

Once missing values are imputed using *K*-nn, MLP, SOM and EM approaches, pattern classification is performed by an artificial neural network with six hidden neurons, which provides accurate classification results in this synthetic problem. Table 1 shows the obtained classification test results for different percentages of missing data in the first attribute. For each imputation method, its parameter values have been selected using cross-validation. Following the obtained classification results, EM algorithm outperforms clearly the remaining tested imputation methods for all missing data percentages. Edited sets obtained using SOM imputation for high missing data percentages (30, 40%) do not facilitate the classifier design and it provides the worst classification results. In this problem, MLP imputation is a better solution than *K*-nn for high percentages of missing data, whereas, *K*-nn approach classifies better for a low missing data percentage.

9.2 Telugu problem

This problem is an Indian vowel recognition dataset. It is a six-class problem composed of 871 cases with three real attributes. In order to make more clear the effect of inserting missing values, the most relevant classification features have been selected. Here we use mutual information because it has been proven that it is a good measure to compute this relevance for classification problems [83]. Thus, a feature x_i is selected to be incomplete if $I(x_i, t)$ is large being $I(x_i, t)$ the normalized mutual information between x_i and the target class variable t. The mutual information has been computed using a Parzen window approach [83]. Before missing data are inserted, the normalized MI between each input feature and the target class are evaluated: $I(x_1, t) = 0.32$, $I(x_2, t) = 0.42$, and $I(x_3, t) = 0.26$. It is critical to observe that not all the features equally contribute to solve the problem. Considering this fact, missing values (5, 10, 20, 30 and 40%) have been randomly introduced in the most relevant feature (x_2). Input data set are split into two separate sets: a training set (70% of samples) and a test set (remaining 30% samples), maintaining the proportion between the instances of each class. As in the Toy problem, model parameters for imputation approaches are chosen by cross-validation (in particular, stratified tenfold cross validation) criterion using the classification results. Once missing values are imputed, an artificial neural network with 18 hidden neurons (experimentally selected number) is trained to learn the classification task. Table 2 shows the obtained classification error rates using the edited training and test sets provided by each imputation approach.

Table 1 Misclassification error rate (mean ± standard deviation from 20 simulations) in Toy problem after missing values are estimated using *K*-nn, MLP, SOM, and EM imputation procedures

A neural network with six hidden neurons is used to perform the classification stage

Missing data in x_1 (%)	Missing data imputation			
	K-nn	MLP	SOM	EM
5	9.21 ± 0.56	9.97 ± 0.48	9.28 ± 0.84	8.29 ± 0.24
10	10.85 ± 1.06	10.86 ± 0.79	9.38 ± 0.52	9.27 ± 0.54
20	11.88 ± 1.01	11.42 ± 0.44	10.63 ± 0.54	10.78 ± 0.59
30	13.50 ± 0.81	12.82 ± 0.51	13.88 ± 0.67	12.69 ± 0.57
40	14.89 ± 0.49	13.72 ± 0.37	15.55 ± 0.66	13.31 ± 0.56

Table 2 Misclassification error rate (mean ± standard deviation from 20 simulations) in Telugu problem after missing values are estimated using *K*-nn, MLP, SOM, and EM imputation procedures

A neural network with 18 hidden neurons is used to perform the classification stage

Missing data in x_2 (%)	Missing data imputation			
	K-nn	MLP	SOM	EM
5	15.92 ± 1.26	15.84 ± 1.13	16.32 ± 1.13	16.19 ± 0.99
10	16.88 ± 1.16	16.87 ± 1.16	16.97 ± 1.18	16.85 ± 1.03
20	18.78 ± 1.29	19.09 ± 1.29	19.30 ± 1.23	19.23 ± 1.12
30	20.58 ± 1.31	20.76 ± 1.34	22.04 ± 1.01	21.22 ± 1.12
40	22.61 ± 1.30	22.76 ± 1.23	24.06 ± 1.29	23.11 ± 1.37

Table 3 Missing data percentages in each input feature of the sick-thyroid dataset

Thyroid dataset	Input feature				
	x_1 (%)	x_2 (%)	x_3 (%)	x_4 (%)	x_5 (%)
Training	10.14	20.89	6.57	10.61	10.54
Test	8.74	18.93	4.84	9.26	9.26

As it can be observed, *K*-nn and MLP imputation provide similar classification results, enhancing the obtained error rates by SOM and EM imputation. These differences are clearer for higher percentages of missing values. If one imputation technique had to be selected, *K*-nn would be a better choice than MLP, because it is a simple solution and provides better classification results. *K*-nn imputation only requires to compute the distances from the incomplete patterns to training vectors and to choose the *K* parameter, whereas MLP imputation requires to train several neural networks (with different weight initializations for avoiding local minima), and it entails a high computational cost.

9.3 Sick-thyroid problem

The information in this dataset comes from thyroid disease records supplied by the Garavan Institute and Quinlan [59]. This binary-classification problem can be found in the UCI database. The original dataset is composed of 2,800 training instances, and 972 test instances. There are two possible thyroid diagnoses (classes): sick and negative. Each input vector is composed of 21 binary attributes, and 7 continuous features (patient age and six laboratory test results). From these features, we select five laboratory test results: x_1 is the total thyroxine test (T_4); x_2 is the free thyroxine test (FT_4); x_3 is the triiodothyronine test (T_3); x_4 is the triiodothyronine uptake test (T_3U) and x_5 is thyropropin test (TSH). The dataset has plenty of missing values and all attributes are incomplete. Table 3 shows the missing data percentages in the five input attributes for each dataset (training and test sets).

As in the two previous problems, we test the influence of missing data estimation into the classifier accuracy using *K*-nn, MLP, SOM, and EM imputation approaches. The model parameters for these procedures are selected by cross-validation (tenfold stratified cross-validation) according to the obtained classification results. Medical diagnosis task (target class variable) is modeled using an artificial neural network composed of 20 hidden neurons (experimentally selected number). Once missing values are imputed, 20 weights initializations are done, and the classifier models are trained using the edited sets provided for each imputation method. Misclassification error rates (average results over the 20 simulations) are shown in Table 4. In this problem, *K*-nn provides the best generalization capabilities, whereas, EM algorithm produce worse classification results than the other methods.

10 Conclusions

Missing or incomplete data is a usual drawback in many real-world applications of pattern classification. Data may contain unknown features due to different reasons, e.g., sensor failures producing a distorted or immeasurable value, data occlusion by noise, non-response in surveys. Handling missing data has become a fundamental requirement for pattern classification because an inappropriate missing data treatment may cause large errors or false classification results.

This work reviews the most important missing data techniques in pattern classification, specially focusing on solutions based on machine learning. In general, pattern classification with missing data concerns two different problems, handling missing values and pattern classification. Depending on how both problems are solved, most of the approaches in the literature can be grouped into four different types of approaches. First, the easiest way for dealing with missing data is listwise or casewise deletion, i.e., omitting the cases that have missing data, the loss of information being its main disadvantage. The second kind of methods is imputation, i.e., to estimate and fill in the unknown input data. In this category, we can distinguish between statistical procedures, such as mean imputation or multiple imputation, and machine learning approaches, such as imputation with neural networks. Other well-known approaches are based on the estimation of the input data distribution, the expectation–maximization algorithm

Table 4 Misclassification error rate (mean ± standard deviation from 20 simulations) in sick-thyroid dataset after missing values are estimated using *K*-nn, MLP, SOM, and EM imputation procedures

	Missing data imputation			
	K-nn	MLP	SOM	EM
Misclassification error rate (%)	3.01 ± 0.33	3.23 ± 0.31	3.49 ± 0.35	3.60 ± 0.31

A neural network with 20 hidden neurons is used to perform the classification stage being the one that stands out. Most of these solutions work well in many situations, being a recommended way to handle missing data. The last methods analyzed in this work are machine learning solutions that are able to handle missing values, such as ensemble of networks or fuzzy methods. Research work on this topic is currently growing wider and it is well known how efficient most of the solutions based on machine learning are.

This review paper provides some experimental results for solutions based on missing data estimation, which comprise the most common and well-known approaches. Imputation is widely used because most commercial decision making software cannot handle with incomplete data, and thus, a previous imputation stage is required to perform the classification stage. As it has been shown in the obtained results, there is not a unique solution that provides the better results for each classification domain. Generally, in real-life scenarios, a detailed study is required in order to evaluate which missing data estimation can help to enhance the classification accuracy.

It is evident that the research direction can still be broadened as there are some open questions that need to be answered, such as “can an efficient imputation solution enhance the decision making process?” In classification tasks with missing values, the main objective of an imputation method must be to help to enhance the classification accuracy. In the recent years, some works have been developed in order to provide missing data estimation focused on solving the classification problem, and not on obtaining imputed values which minimize the error between the predictions and the “true” values. Since it remains quite difficult to compare different techniques, some pending questions to be answered are to establish a unified evaluation criteria and an open access repository for incomplete datasets. Aspect of running time and memory requirements have also to be considered for comparison. The requirements for speed vary from one application to the other. Missing values in fast-moving data such as the stock market need to be estimated in the shortest possible time, whereas this requirement is less important for an opinion poll.

The correct choice of a missing data treatment is a hard and complex task, because the performance of a method depends on the classification, i.e., a method can work well with some problems, and in contrast, its results can be inappropriate with other applications. Thus, a previous analysis of the classification problem to be solved is very important in order to select the most suitable missing data treatment.

Acknowledgments This work is partially supported by Ministerio de Educación y Ciencia under grants TEC2005-00992 and TEC2006-13338/TCM, and also by Consejería de Educación y Cultura de Murcia under grant 03122/PI/05.

References

1. Duda RO, Hart PE, Stork DG (2000) Pattern classification. Wiley, New York
2. Ripley BD (1996) Pattern recognition and neural networks. Cambridge University Press, New York
3. Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford
4. Watanabe S (1985) Pattern recognition: human and mechanical. Wiley, New York
5. Jain AK, Duin RPW, Mao J (2000) Statistical pattern recognition: a review. IEEE Trans Pattern Anal Mach Intell 22(1):4–37
6. Little RJA, Rubin DB (2002) Statistical analysis with missing data, 2nd edn. Wiley, New Jersey
7. Schafer JL (1997) Analysis of incomplete multivariate data. Chapman & Hall, Florida
8. Allison PD (2001) Missing data. Sage university papers series on quantitative applications in the social sciences. Thousan Oaks, California
9. Rubin DB (1987) Multiple imputation for nonresponse in surveys. Wiley, New York
10. Wang L, Fan X (2004) Missing data in disguise and implications for survey data analysis. Field Methods 16(3):332–351
11. Nguyen LN, Scherer WT (2003) Imputation techniques to account for missing data in support of intelligent transportation systems applications. Tech. Rep., University of Virginia, USA
12. Lakshminarayan K, Harp SA, Samad T (2004) Imputation of missing data in industrial databases. Eng Appl Artif Intell 11(3):259–275
13. Ji C, Elwalid A (2000) Measurement-based network monitoring: missing data formulation and scalability analysis. IEEE Int Symp Inf Theory, Sorrento, Italy, p 78
14. Halatchev M, Gruenwald L (2005) Estimating missing values in related sensor data streams. In Int Conf Manage Data, pp 83–94
15. Mohammed HS, Stepenosky N, Polikar R (2006) An ensemble technique to handle missing data from sensors. In: IEEE Sens Appl Symp, Houston, Texas, USA, pp 101–105
16. Cooke M, Green P, Crawford M (1994) Handling missing data in speech recognition. Int Conf Spoken Lang Process, pp 1555–1558
17. Parveen S, Green P (2004) Speech enhancement with missing data techniques using recurrent neural networks. In: IEEE ICASSP, vol 1, pp 733–736
18. DiCesare G (2006) Imputation, estimation and missing data in finance. Ph.D. dissertation, University of Waterloo
19. Sharpe IG, Kofman P (2003) Using multiple imputation in the analysis of incomplete observations in finance. J Financ Econ 1(2):216–249
20. Troyanskaya O, Cantor M, Alter O, Sherlock G, Brown P, Botstein D, Tibshirani R, Hastie T, Altman R (2001) Missing value estimation methods for DNA microarrays. Bioinformatics 17(6):520–525
21. Kim H, Golub GH, Park H (2004) Imputation of missing values in DNA microarray gene expression data. In: Proc IEEE Comput Syst Bioinform Conf
22. Liu P, El-Darzi E, Lei L, Vasilakis C, Chountas P, Huang W (2005) An analysis of missing data treatment methods and their application to health care dataset. In: Li X et al (eds) ADMA, LNCS 3584, Springer, pp 583–590
23. Markey MK, Patel A (2004) Impact of missing data in training artificial neural networks for computer-aided diagnosis. In: Proc Int Conf Mach Learn Appl, pp 351–354
24. Proschan MA, McMahon RP, Shih JH, Hunsberger SA, Geller N, Knatterud G, Wittes J (2001) Sensitivity analysis using an

imputation method for missing binary data in clinical trials. J Stat Plan Inference 96(1):155–165

25. Jerez JM, Molina I, Subirats JL, Franco L (2006) Missing data imputation in breast cancer prognosis. In BioMed'06. ACTA Press Anaheim, CA, pp 323–328
26. Batista G, Monard MC (2003) Experimental comparison of K-nearest neighbour and mean or mode imputation methods with the internal strategies used by C4.5 and CN2 to treat missing data. Tech. Rep., University of Sao Paulo
27. Batista G, Monard MC (2002) A study of K-nearest neighbour as an imputation method. In: Abraham A et al (eds) Hybrid Intell Syst, Ser Front Artif Intell Appl 87, IOS Press, pp 251–260
28. Kohonen T (2006) Self-organizing maps, 3rd edn. Springer
29. Samad T, Harp SA (1992) Self-organization with partial data. Netw Computat Neural Syst 3(2):205–212
30. Fessant F, Midenet S (2002) Self-organizing map for data imputation and correction in surveys. Neural Comput Appl 10(4): 300–310
31. Piela P (2002) Introduction to self-organizing maps modelling for imputation—techniques and technology. Res Stat Note Health Care Financ Adm Off Policy Plan Res 2:5–19
32. Sharpe PK, Solly RJ (1995) Dealing with missing values in neural network-based diagnostic systems. Neural Comput Appl 3(2):73–77
33. Nordbotten S (1996) Neural network imputation applied to the Norwegian 1990 population census data. J Off Stat 12:385–401
34. Gupta A, Lam MS (1996) Estimating missing values using neural networks. J Oper Res Soc 47(2):229–238
35. Yoon SY, Lee SY (1999) Training algorithm with incomplete data for feed-forward neural networks. Neural Process Lett 10:171–179
36. Kallin L (2002) Missing data and the preprocessing perceptron. Tech. Rep., Umeaå University
37. Bengio Y, Gingras F (1995) "Recurrent neural networks for missing or asynchronous data. In: Touretzky DS et al (eds) Adv Neural Inf Process Syst 8. MIT Press, pp 395–401
38. Parveen S (2003) Connectionist approaches to the deployment of prior knowledge for improving robustness in automatic speech recognition. Ph.D. dissertation, University of Sheffield
39. Pyle D (1999) Data preparation for data mining. Morgan Kaufmann Publishers Inc., San Francisco
40. Narayanan S, Vian JL, Choi J, El-Sharkawi M, Thompson BB (2002) Set constraint discovery: missing sensor data restoration using auto-associative regression machines. In: Proc Int Jt Conf Neural Netw, Honolulu, pp 2872–2877
41. Chung D, Merat FL (1996) Neural network based sensor array signal processing. In: Proc Int Conf Multisens Fusion Integr Intell Syst, Washington, USA, pp 757–764
42. Marseguerra M, Zoia A (2005) The autoassociative neural network in signal analysis. II. Application to on-line monitoring of a simulated BWR component. Ann Nuclear Energy 32(11):1207–1223
43. Marwala T, Chakraverty S (2006) Fault classification in structures with incomplete measured data using autoassociative neural networks and genetic algorithm. Curr Sci India 90(4):542–548
44. Caruana R (1997) Multitask learning. Ph.D. dissertation, Carnegie Mellon University
45. Silver DL (2000) Selective transfer of neural network task knowledge, Ph.D. dissertation, University of Western Ontario
46. García-Laencina PJ, Figueiras-Vidal AR, Serrano-García J, Sancho-Gómez JL (2005) Exploiting multitask learning schemes using private subnetworks. In: Cabestany J et al (eds) Comput Intell Bioinsp Syst, Lect Notes Comput Sci 3512, Springer, pp 233–240
47. García-Laencina PJ, Serrano J, Figueiras-Vidal AR, Sancho-Gómez JL (2007) Multi-task neural networks for dealing with missing inputs. In: Mira J, Álvarez JR (eds) IWINAC 2007, part I, Lect Notes Comput Sci 4527, Springer, pp 282–291
48. Ghahramani Z, Jordan MI (1994) Supervised learning from incomplete data via an EM approach. In: Cowan JD et al (eds) Adv Neural Inf Process Syst 6, Morgan Kaufmann Publishers Inc., pp 120–127
49. Ghahramani Z, Jordan MI (1994) Learning from incomplete data. Tech. Rep. AIM-1509, Massachusetts Institute of Technology, Cambridge, MA, USA
50. McLachlan GJ, Krishnan T (1997) The EM algorithm and extensions. Wiley, New York
51. Ahmad S, Tresp V (1993) Some solutions to the missing feature problem in vision. In: Adv Neural Inf Process Syst 5, Morgan Kaufmann Publishers Inc., San Mateo, CA, USA, pp 393–400
52. Tresp V, Ahmad S, Neuneier R (1993) Training neural networks with deficient data. In: Cowan JD et al (eds) Adv Neural Inf Process Syst 6. Morgan Kaufmann Publishers Inc., San Francisco, pp 128–135
53. Tresp V, Neuneier R, Ahmad S (1994) Efficient methods for dealing with missing data in supervised learning. In: Tesauro G et al (eds) Adv Neural Inf Process Syst 7, The MIT Press, pp 689–696
54. Williams D, Liao X, Xue Y, Carin L, Krishnapuram B (2007) On classification with incomplete data. IEEE Trans Pattern Anal Mach Intell 29(3):427–436
55. Ramoni M, Sebastiani P (2001) Robust learning with missing data. Mach Learn 45:147–170
56. Krause S, Polikar R (2003) An ensemble of classifiers for the missing feature problem. In: Proc Intl Jt Conf Neural Netw, Portland, USA, pp 553–558
57. Jian K, Chen H, Yuan S (2005) Classification for incomplete data using classifier ensembles. In: Proc Intl Conf Neural Netw Brain, pp 559–563
58. Juszczak P, Duin RPW (2004) Combining one-class classifiers to classify missing data. In: Roli F et al (eds) Mult Classif Syst, Lect Notes Comput Sci 3077, Springer, pp 92–101
59. Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106
60. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann (Series in Machine Learning)
61. Quinlan JR (1989) Unknown attribute values in induction. In: Proc Intl Workshop Mach Learn, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 164–168
62. Webb GI (1998) The problem of missing values in decision tree grafting. In: Proc Aust Jt Conf Artif Intell, Springer, pp 273–283
63. Zheng Z, Low BT (1999) Classifying unseen cases with many missing values. In: Zhong N, Zhou L (eds) Pac Asia Conf Knowl Discov Data Min, Lect Notes Art Intell 1574, Springer, pp 370–374
64. Clark P, Niblett T (1989) The CN2 induction algorithm. Mach Learn 3(4):261–283
65. Ishibuchi H, Miyazaki A, Kwon K, Tanaka H (1993) Learning from incomplete training data with missing values and medical application. In: Proc IEEE Intl Jt Conf Neural Netw, pp 1871–1874
66. Ishibuchi H, Moriola K (1995) Classification of fuzzy input patterns by neural networks. In: Proc IEEE Intl Conf Neural Netw, Perth, WA, Australia, pp 3118–3123
67. Ishibuchi H, Tanaka H (1991) An extension of the BP-algorithm to interval input vectors-learning from numerical data and expert's knowledge. In: Proc IEEE Intl Jt Conf Neural Netw, pp 1588–1593
68. Petit-Renaud S, Denux T (1998) A neuro-fuzzy model for missing data reconstruction. In: Proc IEEE Workshop Emerg Technol, St. Paul, MN, USA
69. Gabrys B (2000) Pattern classification for incomplete data. In: Proc Intl Conf Knowl Based Intell Eng Syst Allied Technol, Brightom, UK, pp 454–457

70. Gabrys B (2002) Neuro-fuzzy approach to processing inputs with missing values in pattern recognition problems. Int J Approx Reason 30(3):149–179
71. Berthold MR, Huber KP (1998) Missing values and learning of fuzzy rules. Intl J Uncertain Fuzzy Knowl Based Syst 6(2):171–178
72. Berthold MR, Huber KP (1997) Missing values and learning of fuzzy rules. In: Proc Workshop Fuzzy Neuro Syst, 1997
73. Nauck D, Kruse R (1999) Learning in neuro-fuzzy systems with symbolic attributes and missing values. In: Proc 6th Intl Conf Neural Inf Process, Perth, WA, Australia, pp 142–147
74. Hathaway RJ, Bezdek JC (2001) Fuzzy C-means clustering of incomplete data. IEEE Trans Syst Man Cybern B Cybern 31(5): 735–744
75. Ichihashi H, Honda K (2005) Fuzzy c-means classifier for incomplete data sets with outliers and missing values. In: Proc Intl Conf Comput Intell Modell Control Autom, IEEE Computer Society, Washington, DC, USA, pp 457–464
76. Sarkar M, Leong TY (2001) Fuzzy k-means clustering with missing values. In: Proc AMIA Annu Symp, pp 588–592
77. Lim CP, Leong JH, Kuan MM (2005) A hybrid neural network system for pattern classification tasks with missing features. IEEE Trans Pattern Anal Mach Intell 27(4):648–653
78. Bhattacharyya C, Shivaswamy PK, Smola AJ (2004) A second order cone programming formulation for classifying missing data. In: Saul LK et al (eds) Adv Neural Inf Process Syst 17. MIT Press, Cambridge, pp 153–160
79. Smola AJ, Vishwanathan S, Hofmann T (2005) Kernel methods for missing variables. In: Ghahramani Z, Cowell R (eds) Proc AISTATS’05. Society for artificial intelligence and statistics, pp 325–332
80. Pelckmans K, Brabanter JD, Suykens JAK, Moor BD (2005) Handling missing values in support vector machine classifiers. Neural Netw 18(5–6):684–692
81. Bi J, Zhang T (2005) Support vector classification with input data uncertainty. In: Saul LK et al (eds) Adv Neural Inf Process Syst 17. MIT Press, Cambridge, pp 161–168
82. Chechik G, Heitz G, Elidan H, Abbeel P, Koller D (2007) Max-margin classification with incomplete data. In: Schölkopf B et al (eds) Adv Neural Inf Process Syst 19. MIT Press, Cambridge, pp 233–240
83. Kwak N, Choi C-H (2002) Input feature selection by mutual information based on Parzen window. IEEE Trans Pattern Anal Mach Intell 24(12):1667–1671