Neural Processing Letters (2019) 50:2345–2372
https://doi.org/10.1007/s11063-019-10012-0

Artificial Neural Networks with Random Weights for Incomplete Datasets

Diego P. P. Mesquita[1] · João Paulo P. Gomes[2] · Leonardo R. Rodrigues[3]

Published online: 6 March 2019

Abstract
In this paper, we propose a method to design Neural Networks with Random Weights in the presence of incomplete data. We present a method, under the general assumption that the data is missing-at-random, to estimate the weights of the output layer as a function of the uncertainty of the missing data estimates. The proposed method uses the Unscented Transform to approximate the expected values and the variances of the training examples after the hidden layer. We model the input data as a Gaussian Mixture Model with parameters estimated via a maximum likelihood approach. The validity of the proposed method is empirically assessed under a range of conditions on simulated and real problems. We conduct numerical experiments to compare the performance of the proposed method to the performance of popular, parametric and non-parametric, imputation methods. By the results observed in the experiments, we conclude that our proposed method consistently outperforms its counterparts.

Keywords Neural networks · Missing data · Unscented transform

1 Introduction

Artificial Neural Networks (ANN) are a broad class of mathematical models, inspired by the neural mechanism of animals, that have been extensively investigated in the past decades. Successful applications of ANN have been reported in various domains such as signal processing [47], control systems [36] and computer vision [51].

Among the aspects that influence the performance of an ANN, the learning method is one of the most relevant. In this context, gradient-based learning algorithms have played a dominant

✉ João Paulo P. Gomes
jpaulo@lia.ufc.br

Diego P. P. Mesquita
diego.mesquita@aalto.fi

Leonardo R. Rodrigues
leonardolrr2@fab.mil.br

[1] Department of Computer Science, Aalto University, Espoo, Finland

[2] Department of Computer Science, Federal University of Ceará, Fortaleza, CE, Brazil

[3] Electronics Division, Institute of Aeronautics and Space, São José dos Campos, SP, Brazil

role in training ANN [45]. However, problems such as slow convergence and convergence to local minima increased the popularity of Neural Networks with Random Weights (NNRW). NNRWs provide a solution to these issues by randomly assigning the hidden weights and adjusting only the output neurons during the training step. In one of the first works to analyze NNRW, [43] reported that the weights of the output layer are the most relevant, and the other weights may not need to be tuned once they are properly initialized. The authors argue that randomly setting the parameters in the hidden neurons helps to remove the redundancy of the model in parameter space and thus makes the performance less sensitive to the resulting parameters compared with other typical learning rules such as back-propagation. Similar basic ideas were also reported in the seminal work of Broomhead and Lowe [3], where the authors showed that the random selection of centers among the training data is a valid alternative to sophisticated clustering algorithms used in RBF neural networks.

Since the 90s, NNRWs have become increasingly popular and various models were proposed. For example, in [37], the authors proposed a new NNRW model named Random Vector Functional Link (RVFL). The RVFL model differs from the model presented by Schmidt et al. [43] since it includes a direct link between the inputs and the output layer. Later, several authors showed that the direct link may enhance the performance of RVFL [46]. Furthermore, various authors such as [2,14,15] introduced NNs with a randomly initialized hidden layer, trained using the pseudo-inverse. The universal approximation capability of NNRWs was also addressed in works such as [20,26,40]. More recently, several authors introduced NNRWs with different characteristics such as constructive/selective NNs [49] and deep architectures [13,48]. Such models achieved remarkable performances and may point promising future research directions. It is worth pointing that the literature of NNRW is vast and, in this paper, we do not aim to present it in details. For a more complete survey on NNRW including historical perspectives and current challenges, the reader may refer to [42,52].

Despite the vast number of successful applications [5,39,41], the use of NNRW can be limited by the presence of observations with one or more missing values. The occurrence of missing values can be caused by many reasons such as measurement error, device malfunction and operator failure. To overcome the problems derived from the occurrence of these missing components, it is necessary to understand the mechanisms governing this phenomenon. According [29], the missingness mechanism can be characterized as Not Missing at Random (NMAR), Missing Completely at Random (MCAR) and Missing at Random (MAR).

The NMAR mechanism describes the situation in which the probability of a component being missing is related to the value of this component. If missing data are NMAR, building a missingness model is the only way to obtain unbiased estimates of the missing values. In MCAR, the missingness of a component is independent of its real value or any value of other components on the dataset. Such a characterization is very restrictive, and it makes the MCAR case unlikely in real life problems [8]. A more reasonable assumption states that the data are MAR. In the MAR mechanism, the missingness of a component is independent of the value itself but can be related to the observed values. Under the assumption that the data are MAR, strategies for dealing with missing data in machine learning methods (including neural networks) can be grouped into three different approaches: deletion of incomplete cases, imputation of missing values and design of learning methods that can handle missing data directly.

The deletion approach is, by far, the most common [50]. In the listwise deletion, instances with at least one missing attribute are eliminated, and the analysis is performed using the rest of the examples. Although commonly used, listwise deletion can lead to a performance loss when the number of instances with missing attributes is significant [12].

The imputation approach consists of strategies for filling the missing values using the information available from the observed values of the dataset. These methods can be split into single and multiple imputation procedures. In single imputation, each missing value is filled with a single estimated value. Examples of single imputation methods are the incomplete case nearest neighbors imputation (ICk-NNI, [17]), where the value is imputed according to the k-nearest neighbors obtained with an incomplete distance metric and the Expectation Conditional Maximization (ECM, [31]). In the ECM method, a parametric model is assumed for the dataset, and the EM algorithm is used to estimate the parameters of the distribution. After that, the missing values are imputed with its expected value conditioned to the observed values of the instance. After the imputation procedure, a standard learning algorithm can be applied.

In contrast to single imputation, multiple imputation methods attribute a set of possible values rather than a single value for the missing attributes. Hence, these methods generate some different datasets where the complete instances are identical, but the incomplete instances have different values [23]. After that, learning algorithms are trained on each dataset, and a final result is obtained by combining all generated models. It is worth noting that multiple imputation methods present a significant increase in computational cost when compared to single imputation methods. However, multiple imputation approaches can handle the inherent uncertainty from the missing value estimation process in the final machine learning model thus leading to better results.

The third group of strategies consists of the adaptation of learning methods to handle missing data on its formulation. Such methods avoid direct imputation procedures [12]. In [9], the authors propose a method to estimate the expected value of pairwise distances between vectors with missing data using a Gaussian Mixture Model (GMM) for the distribution of the dataset. The proposed method is used to design a k-NN classifier for datasets with missing values. Support Vector Machines (SVM) and Gaussian Process (GP) variants for missing data were proposed in [44], where both SVM and GP are cast in a general framework for kernel methods as an estimation problem using exponential families in feature space. Also for SVM, the work in [38] propose a modified risk function that incorporates a probability distribution for missing data. Mesquita et al. [33] presents a method to estimate the expected value of the Gaussian kernel calculated over incomplete vectors.

Considering NNRWs, methods for directly handling missing data are indeed rare. Most of the works available in the literature are related to the use of neural networks for single imputation, as can be seen in [1,19]. Other authors propose single imputation methods [35] or evaluate the impact of various well-known imputation methods on the performance of NNRW [30]. It is worth noting that these single imputation methods were not particularly designed to work with NNRW and can be used with almost any machine learning method.

In this paper, we propose a method to design NNRW models that can handle missing components directly. In the proposed framework, we consider each entry as a random variable with its distribution modeled with a GMM. We use the Unscented Transform (UT) to estimate both the expected value and the variance of each entry after going through the hidden layer. Then, we estimate the weights of the output layer using an uncertainty robust least squares formulation. We tested our method on two NNRW models in synthetic and real datasets. Results show that our approach outperformed other commonly used missing data treatments, especially when the number of missing components is high. The proposed method differs from other missing data treatments because it accounts for the uncertainty of the imputation process, similar to multiple imputation methods, whereas it does not rely on a computationally intensive sampling procedure. In this way, our method can be seen as an alternative to single and multiple imputation methods, since it provides more accurate

estimates than single imputation methods (since it considers the uncertainty) with a reduced computational cost when compared to multiple imputation procedures.

The remainder of this paper is organized as follows. Section 2 presents a brief review of basis functions and NNRW models. Section 3 describes the proposed method to design NNRW models that are capable of handling missing values. Section 4 summarizes the steps to be followed to implement the proposed method. Section 5 presents the results obtained in numerical experiments conducted to evaluate the performance of the proposed method. Concluding remarks are given in Sect. 6.

2 Basis Expansion and Neural Networks with Random Weights

Let $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ be a training set with N examples, such that $\mathcal{X} = \{X_i\}_{i=1}^{N}$ and $\mathcal{Y} = \{Y_i\}_{i=1}^{N}$ are, respectively, a set of D-dimensional input points and their corresponding outputs. Assuming the existence of a continuous mapping $\mathfrak{f} : \mathcal{X} \to \mathcal{Y}$ between the input and the output space, we want to estimate $\mathfrak{f}$ from data. In many NNRW models, $\mathfrak{f}$ can be expressed as a linear combination of nonlinear functions (basis functions) over the input variables as follows:

$$\hat{Y}_i = \sum_{j=0}^{\mathcal{M}} \beta_j \phi_j(X_i), \tag{1}$$

where the function $\phi_j(\cdot) : \mathbb{R}^D \to \mathbb{R}$ is the j-th basis function, β_j is its corresponding weight and $\mathcal{M}$ is the number of hidden neurons. Usually $\phi_0(\cdot) = 1$ and β_0 is the bias term.

The resulting model, referred as linear basis expansion, can approximate a nonlinear function while still linear in the parameters. Under the condition that these basis functions are infinitely differentiable, this model can approximate any given function ([6,11,16]). Since the model is linear concerning the basis functions outputs, the parameters are typically estimated with maximum likelihood or Bayesian techniques.

The use of different basis functions defines various neural networks with random weights models such as the Feedforward Neural Network with Random Weights (FNNRW, [43]), the Random Vector Functional Link (RVFL, [36]) and the q-Generalized Random Neural Network (QGRNN, [45]) among others. To illustrate our point, we present reformulated versions of FNNRW and QGRNN according to the basis expansion model. Similar procedures can be performed for other NNRW models and require only straightforward mathematical manipulations.

The FNNRW model, originally proposed in [43], was one of the first to introduce the concept of NNRW. The FNNRW comprises a multilayer perceptron model where the weights between the input and the hidden neurons are randomly assigned. Such model can be written as:

$$\hat{Y}_i = \sum_{j=0}^{\mathcal{M}} \beta_j f(\boldsymbol{w}_j^T X_i + c_j) \tag{2}$$

where $\boldsymbol{w}_j$ is the weight vector connecting the input layer and the j-th hidden neuron, β_j is the weight connecting the j-th hidden neuron and output layer, c_j is the threshold of the j-th hidden neuron and $f(\cdot)$ is the logistic sigmoid function. As can be noticed, in the FNNRW model the basis function is defined as a sigmoid function over the linear combination of the input vector.

The QGRNN model can be written similarly. The q-Generalized Random Neural Network is a recently proposed NNRW that uses radial basis neurons and a q-Gaussian activation function. The resulting model can be expressed as:

$$\hat{Y}_i = \sum_{j=0}^{\mathcal{M}} \beta_j g(X_i) \tag{3}$$

where:

$$g(\alpha) = e_q(-||\alpha - \gamma||^2/\eta^2) \tag{4}$$

where γ and η represent, respectively, the center and the width of the q-Gaussian, and e_q is the q-exponential defined by:

$$e_q(\alpha) = [1 + (1-q)\alpha]^{\frac{1}{1-q}} \tag{5}$$

The choice of q, the entropic index, is of fundamental importance since it results in different deformations of the Gaussian distribution. Once again, the NNRW model can be expressed as a linear basis expansion model.

3 Neural Networks with Random Weights for Datasets with Missing Values

Given a set of possibly incomplete training inputs $\mathcal{X}$ and the set of corresponding outputs $\mathcal{Y}$, we wish to estimate the weight vector $\mathcal{B} = (\beta_0, \ldots, \beta_{\mathcal{M}})^T$. To do so, we model the missing entries as random variables and choose $\mathcal{B}$ to minimize the expected sum of square errors loss function. We derive a solution in terms of the expected values and the covariance matrices of the transformed inputs. In turn, we provide a method to estimate these statistics given the distribution of the input data. Furthermore, we consider that missing values are MAR [29] and that the full version of the examples in $\mathcal{X}$ are i.i.d.

3.1 Formulation

Given the aforementioned assumptions, each $\phi_j(X_i)$ is an univariate random variable since it is a transform of X_i for which several components may be missing. In such scenario, $\mathcal{B}$ can be estimated by solving the following optimization problem:

$$\underset{\mathcal{B}}{\text{minimize}} \quad \mathbb{E}\left[\sum_{i=1}^{N}\left(Y_i - \sum_{j=0}^{\mathcal{M}} \beta_j \phi_j(X_i)\right)^2\right] \tag{6}$$

For a compact notation, let us denote $\psi_i = (\phi_0(X_i), \ldots, \phi_{\mathcal{M}}(X_i))^T$, $\Psi = (\boldsymbol{\psi_1}, \ldots, \boldsymbol{\psi_N})^T$, $\boldsymbol{Y} = (Y_1, \ldots, Y_N)^T$ and $\mathcal{B} = (\beta_0, \ldots, \beta_{\mathcal{M}})^T$. Thus, the same problem can be expressed as:

$$\underset{\mathcal{B}}{\text{minimize}} \quad \mathbb{E}\left[||\boldsymbol{Y} - \Psi\mathcal{B}||_2^2\right] \tag{7}$$

We can assume that Ψ is a random variable with $\overline{\Psi} = \mathrm{E}[\Psi]$, so we can describe Ψ as $\Psi = \overline{\Psi} + U$, where U is a random matrix with zero mean. By doing so, the objective function in Eq. (7) can be expressed as:

$$\begin{aligned}\mathbb{E}\left[||\boldsymbol{Y}-\Psi\mathcal{B}||_2^2\right] &= \mathbb{E}\left[(\boldsymbol{Y}-\overline{\Psi}\mathcal{B}-U\mathcal{B})^T(\boldsymbol{Y}-\overline{\Psi}\mathcal{B}-U\mathcal{B})\right] \\ &= (\boldsymbol{Y}-\overline{\Psi}\mathcal{B})^T(\boldsymbol{Y}-\overline{\Psi}\mathcal{B}) + \mathrm{E}\left[\mathcal{B}^T U^T U\mathcal{B}\right] \\ &= ||\boldsymbol{Y}-\overline{\Psi}\mathcal{B}||_2^2 + \mathcal{B}^T P\mathcal{B}\end{aligned} \tag{8}$$

where $P = \mathrm{E}[U^T U]$. Therefore, the problem has the form of a regularized least squares problem and the solution is given by:

$$\mathcal{B} = (\overline{\Psi}^T\overline{\Psi} + P)^{-1}\overline{\Psi}^T\boldsymbol{Y} \tag{9}$$

We can express P as a function of the moments of the random variable Ψ. For any $l, l' \in \{1, \ldots, \mathcal{M}\}$, the element $P_{l,l'}$, in the intersection of the l-th line with the l'-th column of P is given by:

$$\begin{aligned}P_{l,l'} &= \sum_{i=1}^{N}\mathrm{E}[U_{i,l}U_{i,l'}] \\ &\triangleq \sum_{i=1}^{N}\Big(\mathrm{Cov}(U_{i,l}, U_{i,l'}) + \mathbb{E}[U_{i,l}]\mathbb{E}[U_{i,l'}]\Big) \\ &= \sum_{i=1}^{N}\mathrm{Cov}(U_{i,l}, U_{i,l'}) \\ &= \sum_{i=1}^{N}\mathrm{Cov}(\Psi_{i,l}, \Psi_{i,l'}).\end{aligned}$$

According to this result, it follows directly that:

$$P = \sum_{i=1}^{N}\mathrm{Cov}(\psi_i) \tag{10}$$

Consequently, estimating P consists in computing the covariance matrices of the projected input vectors $\psi_1, \ldots, \psi_N$ and summing over them. According to the results presented in Eqs. (9) and (10), we have to estimate $\mathbb{E}[\psi]$ and $\mathrm{Cov}(\psi)$ to obtain $\mathcal{B}$.

3.2 Computing $\mathbb{E}[\boldsymbol{\psi}]$ and $\mathbf{Cov}(\boldsymbol{\psi})$

Let $X \in \{X_1, \ldots, X_N\}$ be an input vector and ψ its projected version. The problem of computing the expectation of ψ consists in evaluating individually the expectation of its entries $\phi_1(X), \ldots, \phi_{\mathcal{M}}(X)$, given by:

$$\mathbb{E}[\phi_l(X)] = \int_{\mathbb{R}^D}\phi_l(X)p(X)dX \quad \forall l \in \{1, \ldots, \mathcal{M}\}, \tag{11}$$

while the elements of the covariance matrix $\mathrm{Cov}(\psi)$ are given by:

$$\begin{aligned}\mathrm{Cov}(\phi_l(X), \phi_{l'}(X)) = \int_{\mathbb{R}^D}(\phi_l - \mathbb{E}[\phi_l])(\boldsymbol{\phi}_{l'} - \mathbb{E}[\phi_{l'}])p(X)dX \\ \forall l, l' \in \{1, \ldots, \mathcal{M}\}\end{aligned} \tag{12}$$

for which there is no trivial general solution and tailored ones depend on both the format of $\phi(\cdot)$ and $p(\cdot)$. However, for any $\phi(\cdot)$ and $p(\cdot)$, it is possible to approximate Eqs. (11) and (12) via sampling or numerical integration methods.

The Unscented Transform (UT), initially proposed in [21], is a sampling-based method that estimates the statistical moments of a probability distribution associated with a random variable which results from a nonlinear transformation of another random variable [24]. Here, we wish to estimate the moments of ψ, which results from the application of $\phi_1(\cdot), \ldots, \phi_{\mathcal{M}}(\cdot)$ on the random variable X.

The UT pipeline consists of (1) selecting a set of samples from the original distribution. We consider the sampling scheme proposed by Julier and Uhlmann [22], where a set of $L = 2|M| + 1$ samples is deterministically chosen, where $|M|$ is the number of missing entries of X. Additionally, we compute a set of weights that will be later used to retrieve the moments of the transformed variable. (2) Evaluating the values of $\phi_1(\cdot), \ldots, \phi_{\mathcal{M}}(\cdot)$ on the previously selected samples. (3) Estimate the moments of the transformed variable ψ using the transformed samples obtained in step 2 and the weights from step 1. Further details regarding the UT are presented in the Appendix.

Although this UT approach could be applied directly to estimate the statistical moments of an arbitrary basis function, the number of required samples grows with the number of missing entries, increasing the computational burden of the procedure. To alleviate this problem, we propose two methodologies based on the UT that require only three one-dimensional sigma points, independent of $|M|$. The first one, presented in Sect. 3.2.1, is tailored to the logistic function and can be easily generalized to any $\phi(\cdot)$ that can be expressed as a transform of $\boldsymbol{w}^T X$. The second one, presented in Sect. 3.2.2, deals with the q-Gaussian function and can be adapted for any function that is a transform of $\|\boldsymbol{w}^T X\|^2$. In this paper, we consider that $\mathrm{Cov}(\psi_i), \ldots, \mathrm{Cov}(\psi_N)$ are diagonal matrices. Thus, to obtain an approximation of P, it suffices to compute the individual variances of the basis functions evaluated at each $X \in \{X_i, \ldots, X_N\}$.

3.2.1 Sigmoid Function

Given an input vector X, the sigmoid function is given by:

$$f(X) = \frac{1}{1 + e^{-\boldsymbol{w}^T X}} \tag{13}$$

where $\boldsymbol{w} = (w_1, \ldots, w_D)^T$ is a predefined constant vector.

Note that $f(X)$ can be written as a transform of the random variable $\boldsymbol{w}^T X$, whose expectation is given by:

$$\mathbb{E}[\boldsymbol{w}^T X] = \sum_{d=1}^{D} w_d \mathbb{E}[x_d], \tag{14}$$

and has variance:

$$\mathrm{Var}[\boldsymbol{w}^T X] = \sum_{d=1}^{D} w_d^2 \mathrm{Var}[x_d]. \tag{15}$$

Using the UT scheme (see details in Appendix), we can approximate $\mathbb{E}[f(X)]$ with $L = 3$ sigma points as follows:

$$\mathbb{E}[f(X)] \approx \sum_{i \in \{-1,0,1\}} \frac{u_i}{3} \tag{16}$$

where:

$$u_i = \left(1 + \exp\{-\mathbb{E}[\boldsymbol{w}^T X] + i\sqrt{\mathrm{Var}[\boldsymbol{w}^T X]}\}\right)^{-1} \tag{17}$$

Analogously, the variance of the logistic function $f(X)$ is given by:

$$\text{Var}(f(X)) \approx \sum_{i\in\{-1,0,1\}} \frac{1}{3}\left(u_i - \mathbb{E}[f(X)]\right)^2 \tag{18}$$

It is important to notice this methodology also applies to any transform of X that can be written as a function of $\boldsymbol{w}^T X$.

3.2.2 q-Gaussian Function

Given an input vector X, the q-Gaussian activation function can be expressed as:

$$g(X) = e_q(-\|X - \boldsymbol{w}\|^2 \nu^{-1}) \tag{19}$$

where $\boldsymbol{w} = (w_1, \ldots, w_D)^T$, $\nu > 0$ and $q \in \mathbb{R}$ are predefined constants while

$$e_q(\alpha) = [1 + (1-q)\alpha]^{\frac{1}{1-\alpha}}. \tag{20}$$

Note that $g(X)$ can be written as a transform of $\|X - \boldsymbol{w}\|^2$, whose expectation is given by

$$\mathbb{E}[\|X - \boldsymbol{w}\|^2] = \sum_{d=1}^{D} (\mathbb{E}[x_d] - w_d)^2 + \text{Var}[x_d], \tag{21}$$

and has variance:

$$\text{Var}[\|X - \boldsymbol{w}\|^2] = \sum_{d=1}^{D} \mathbb{E}[x_d^4] - \mathbb{E}[x_d^2]^2 + 4w_d^2 \text{Var}[x_d]. \tag{22}$$

Thus, $\mathbb{E}[g(X)]$ can be approximated using the aforementioned UT scheme with $L = 3$ sigma points, as follows:

$$\mathbb{E}[g(X)] \approx \sum_{i\in\{-1,0,1\}} \frac{r_i}{3} \tag{23}$$

where

$$r_i = e_q(-(\mathbb{E}[\|X - \boldsymbol{w}\|^2] + i\sqrt{\text{Var}[\|X - \boldsymbol{w}\|^2]})\nu^{-1}) \tag{24}$$

Analogously, the variance of the q-Gaussian activation function $g(X)$ is given by:

$$\text{Var}(g(X)) \approx \sum_{s\in\{-1,0,1\}} \frac{1}{3}\left(r_i - \mathbb{E}[g(X)]\right)^2 \tag{25}$$

Notice that this methodology can be trivially adapted to estimate the value of any specific transform $g(X)$ that can be expressed as a function of $\|X - \boldsymbol{w}\|^2$.

3.3 Modeling the Data with a Gaussian Mixture Distribution

The developments presented so far require the computation of the expected values and variances of the missing components in each feature vector. For this purpose, we modeled the distribution of the input data with a Gaussian Mixture. Then, we condition this distribution on the observed features, obtaining the statistical moments of interest.

In order to provide a flexible representation for the distribution from which the feature vectors were drawn, we assume that it is possible to model the distribution as a linear superposition of C D-dimensional Gaussian densities. Each Gaussian density has mean $\mu^{(c)}$ and covariance matrix $\Sigma^{(c)}$, with $c = [1, \ldots, C]$, i.e. a Gaussian Mixture Model (GMM). Given an arbitrary $X_i \in \mathbb{R}^D$, the probability density function of a GMM with the aforementioned parameters takes the form [18]:

$$p(X_i) = \sum_{c=1}^{C} w^{(c)} \mathcal{N}(X_i | \mu^{(c)}, \Sigma^{(c)}), \tag{26}$$

where $\{w^{(c)}\}_{c=1}^{C}$ is a set of non-negative scalars that satisfies the constraint $\sum_{c=1}^{C} w^{(c)} = 1$. The GMM model is a flexible and powerful modeling tool capable to model a wide range of continuous distributions, provided a sufficient number of Gaussian components. The parameters $\{w^{(c)}, \mu^{(c)}, \Sigma^{(c)}\}_{c=1}^{C}$ of the GMM can be estimated via Maximum Likelihood using Expectation Maximization [31].

Consider an arbitrary vector X_i, with missing component values $X_{i,M}$ and observed component values $X_{i,O}$, where M and O denote the sets of indexes of missing and observed component values, respectively. Then, the parameters $\mu^{(c)}$ and $\Sigma^{(c)}$ of the c-th component of the GMM can be partitioned into two blocks as follows:

$$\mu^{(c)} = \begin{bmatrix} \mu_O^{(c)} \\ \mu_M^{(c)} \end{bmatrix}, \quad \Sigma^{(c)} = \begin{bmatrix} \Sigma_{OO}^{(c)} & \Sigma_{OM}^{(c)} \\ \Sigma_{MO}^{(c)} & \Sigma_{MM}^{(c)} \end{bmatrix}. \tag{27}$$

Then, the mean vector $\tilde{\mu}_i^{(c)} = \mathrm{E}^{(c)}[X_{i,M}|X_{i,O}]$ and the covariance matrix of the c-th Gaussian in the GMM, conditioned on $X_{i,O}$, $\tilde{\Sigma}_i^{(c)} = \mathrm{Var}^{(c)}[X_{i,M}|X_{i,O}]$, are given by:

$$\tilde{\mu}_i^{(c)} = \mu_M^{(c)} + \Sigma_{MO}^{(c)} (\Sigma_{OO}^{(c)})^{-1} (X_{i,O} - \mu_O^{(c)}), \tag{28}$$

$$\tilde{\Sigma}_i^{(c)} = \Sigma_{MM}^{(c)} - \Sigma_{MO}^{(c)} (\Sigma_{OO}^{(c)})^{-1} \Sigma_{OM}^{(c)}, \tag{29}$$

and the expected value and the covariance matrix of the missing features $X_{i,M}$ are given by:

$$\mathbb{E}[X_{i,M}] = \sum_{c=1}^{C} w^{(c)} \tilde{\mu}_i^{(c)}, \tag{30}$$

$$\mathrm{Cov}(X_{i,M}) = \sum_{c=1}^{C} w^{(c)} (\tilde{\Sigma}_i^{(c)} + \tilde{\mu}_i^{(c)} \tilde{\mu}_i^{(c)^T}) - \mathbb{E}[X_{i,M}] \mathbb{E}[X_{i,M}]^T. \tag{31}$$

4 Implementation

Given a dataset $\mathcal{X} = \{X_i\}_{i=1}^{N} \subset \mathbb{R}^D$ with possibly incomplete feature vectors, we wish to compute the weights for a NNRW model. The proposed method can be briefly described by the following steps:

1. Estimate the parameters of the Gaussian mixture distribution of the data $\mathcal{X}$ with C components (mean vectors $\mu^{(c)}$, covariance matrices $\Sigma^{(c)}$ and coefficients $w^{(c)}$, with $c = 1 \ldots, C$) by maximizing the likelihood function of the mixture model by Expectation-Maximization adapted for incomplete data [18,31];

2. Compute the mean vectors $\tilde{\mu}_i^{(c)}$ and the covariance matrices $\tilde{\Sigma}_i^{(c)}$ of the missing components $X_{i,M}$ for each of the $c = 1, \ldots, C$ components, using Eqs. (28) and (29);
3. Compute the expected value and the covariance matrix $X_{i,M}$, using Eqs. (30) and (31);
4. Compute the expected value $\mathbb{E}[\psi]$ and the covariance matrix $\text{Cov}(\psi)$ of the transformed input vectors, as described in Sect. 3.2;
5. Compute $\mathcal{B}$ using Eq. (9).

5 Experiments and Results

In order to assess the performance of the proposed framework, we conducted experiments with the logistic and the q-Gaussian basis functions as well as the NNRW models based on both basis functions (FNNRW and QGRNN). The objective of our experiments is twofold: verify how well our proposal can approximate the value of a basis function over an incomplete vector and assess the performance of NNRW models that use our framework. All experiments are described in details in the following subsections.

5.1 Multivariate Normal Data with Known Parameters

In this experiment, we want to estimate the q-Gaussian and the sigmoid activation functions computed over a vector with missing components. The vectors were drawn from a 5-dimensional multivariate normal distribution with known mean and covariance matrix. We varied the number of missing components. The design of an experiment with a known distribution aims at verifying the effect of the number of missing components in our proposal without the influence of the distribution estimation algorithm (i.e. GMM).

Initially, we randomly selected the parameters that will generate the tested vectors. For that, the mean vector was sampled from the standard normal distribution $N(0, 1)$. The covariance matrix is given by $\Sigma_s = L^T L$, where L is an upper triangular matrix whose non-zero entries were also drawn from the same distribution $N(0, 1)$. The procedure to generate those parameters was repeated 20 times and the results reported in this section are the average of these 20 rounds.

Given the mean μ_s and covariance matrix Σ_s, we created a dataset with 10^3 samples drawn from this Normal distribution. We gradually increased the number of missing entries and verified the impact on the performance of our method as well as on two other common imputation methods. At each round, the entries of $\boldsymbol{w}$, used for both the q-Gaussian and the sigmoid functions, were drawn at random from a standard normal $N(0, 1)$.

Our proposal was compared to the Single Mean Imputation (SMI), that imputes all missing components with the average value calculated using the complete components and the Conditional Mean Imputation (CMI, [31]). In CMI, the missing entries are imputed with their expected values conditioned to the observed entries of the same vector. Then, we calculated the basis function using the imputed values. For all methods, the data distribution was considered to be known. Since all methods share the same statistical model, the performance of each one depends only on its formulation.

Tables 1 and 2 show, respectively, the Average Root Mean Squared Errors (ARMSE) between real and estimated basis functions computed for the logistic function and the q-Gaussian function.

By the results, we observe that the performance of all methods degrades as the number of missing entries increases. We also conclude that all methods have similar performances

Table 1 ARMSE for the logistic function

%	SMI	CMI	UT
20	0.017475	0.001142	0.001119
40	0.037474	0.012283	0.011103
60	0.070443	0.041540	0.036681
80	0.111060	0.085178	0.073529

Table 2 ARMSE for the q-Gaussian function

%	SMI	CMI	UT
20	0.1219	0.0031	0.0030
40	1.0684	0.2342	0.2153
60	2.5528	1.2162	1.0932
80	11.1290	4.7209	4.2292

for the lowest missingness levels. In such cases, the uncertainty on the imputation procedure seems to have a small impact on the estimate of the basis functions. It is worth noting that our method outperforms all other methods for the highest missingness levels. This result is expected since the high number of missing components increases the uncertainty of the estimation process and our method is the only one that includes this factor in its formulation.

5.2 Experiments Using Real-World Data

We evaluate the performance of the proposed method using real-world datasets. For that purpose, eleven datasets (see Table 3) were selected from the UCI Machine Learning Repository [28].

For both the logistic and the q-Gaussian functions, twenty similar rounds of the experiment were carried out. In each of these rounds, we selected 80% of the dataset for training and 20% for testing. The percentage of instances with missing samples was interactively increased from 10 to 50% (in steps of 10%) of the dataset size. In each step, we used different methods to estimate the basis function and measured the ARMSE between the obtained estimates and the real function values computed beforehand. Since the real distribution of the data is unknown, we estimate a GMM comprising three components.

Besides SMI and CMI, we also compare our method against the Incomplete-Case k-Nearest-Neighbors Imputation algorithm (ICkNNI), [17] and the Singular Value Thresholding (SVT) [4]. ICkNNI is a popular imputation method due to its easiness of implementation and good performance. Unlike most of single imputation strategies, ICkNNI does not rely on the estimation of a statistical model for the data. For ICkNNI, we used the parameters suggested in [17]. SVT is a matrix factorization based imputation method that has been largely used in several real-world problems such as movie recommendation and image recovery [27]. All tests were performed using the SVT toolbox described in [25].

Tables 4 and 5 show the results, in terms of ARMSE, obtained for the logistic function and the q-gaussian, respectively. The comparisons between the proposed method, SMI and CMI lead to conclusions that are similar to the ones obtained in previous experiments. The UT-based method outperforms SMI and CMI when the number of missing components increases. It is important to point out that UT was also able to outperform ICkNNI and SVT in most of the datasets. This result is significant since ICkNNI and SVT do not rely on any assumption

Table 3 Datasets description

	# Features	# Samples
Cancer	194	32
MPG	392	7
CPU	209	9
Concrete compression	1030	80
Boston housing	506	13
Red wine	1599	11
White wine	4898	3
Diabetes	768	8
Monk 1	556	6
Monk 2	601	6
Monk 3	554	6

regarding the distribution of the dataset and UT used a statistical model with a fixed, and possibly non-optimal, number of Gaussians.

Aiming at assessing the statistical significance of our results, we conducted a Friedman statistical test [10]. The Friedman test quantifies the consistency of the results obtained by a method when applied in several datasets. The success of each method is quantified according to their performance rankings (for each dataset, the best performing algorithm getting the rank of 1, the second best rank 2 and so on). In the current setting, the null hypothesis H_0 states that there is no statistical difference between all models.

Following the methodology presented in [7], we verify that the null hypothesis was rejected with $p = 5.3051e-23$ for the logistic and $p = 9.0224e-16$ for the q-Gaussian. We obtained a Critical Difference (CD) of $CD = 0.82251$. By observing the CD and the average rankings, we can state that our method significantly outperformed all other methods for the logistic activation. For the q-Gaussian, our proposal showed a significant performance gain when compared to SVT and SMI. It is worth pointing that, even though the performance gap between UT and the other two methods (ICkNNI and CMI) was not significant, our proposal had the best average ranking.

5.3 Application to Neural Networks with Random Weights

In this last experiment, our goal is to verify the impact of the basis function estimates on NNRW models. The proposed variants of FNNRW and QGRNN were compared to its original formulations trained with imputed datasets using SMI, CMI, SVT and ICkNNI. All FNNRW and QGRNN models have 100 hidden neurons.

In this experiment, the number of missing components was iteratively increased from 10 to 50% (in steps of 10%). SMI, CMI and UT used GMM models comprising three Gaussian. The parameters of each Gaussian was estimated with the ECM algorithm. We repeated the experiments twenty times and computed the ARMSE. For all datasets, we randomly selected 80% for training and 20% for testing. Tables 6 and 7 show the results for FNNRW and QGRNN, respectively.

The results presented in Tables 6 and 7 show that our proposal outperformed all other methods in more than 90% of the tested cases. This experiment enforces the belief that better estimates of basis functions may lead to NNRW with better performance, although no theoretical guarantee could be given. Despite the performance gain, one crucial aspect that

Table 4 Comparison of different approaches to estimate the logistic function in presence of missing entries on real-world data

	%	SVT	ICkNNI	SMI	CMI	UT
Cancer	10	0.0363 ± 0.00409	0.0459 ± 0.00522	0.0693 ± 0.00811	0.037 ± 0.00876	0.0363 ± 0.00859
	20	0.0426 ± 0.00472	0.0679 ± 0.00624	0.099 ± 0.00744	0.0542 ± 0.00916	0.053 ± 0.00893
	30	0.057 ± 0.00441	0.088 ± 0.00773	0.124 ± 0.0082	0.0665 ± 0.00945	0.065 ± 0.00939
	40	0.0651 ± 0.00979	0.103 ± 0.00841	0.141 ± 0.00955	0.0749 ± 0.00956	0.0733 ± 0.00944
	50	0.0757 ± 0.0114	0.115 ± 0.00736	0.157 ± 0.00758	0.0852 ± 0.0126	0.0834 ± 0.0124
MPG	10	0.0397 ± 0.00561	0.0394 ± 0.00483	0.063 ± 0.00505	0.0387 ± 0.00421	0.0377 ± 0.00396
	20	0.0554 ± 0.00397	0.0528 ± 0.00662	0.088 ± 0.00487	0.0549 ± 0.0107	0.0536 ± 0.0102
	30	0.0738 ± 0.00441	0.0661 ± 0.00608	0.109 ± 0.00441	0.0685 ± 0.014	0.0667 ± 0.0128
	40	0.0828 ± 0.00465	0.0796 ± 0.00729	0.126 ± 0.0067	0.0798 ± 0.0104	0.0778 ± 0.00977
	50	0.0948 ± 0.00447	0.0876 ± 0.00699	0.14 ± 0.00551	0.0894 ± 0.012	0.0872 ± 0.0115
Compression	10	0.0429 ± 0.00309	0.0348 ± 0.00508	0.0598 ± 0.00368	0.0403 ± 0.00511	0.0393 ± 0.00481
	20	0.0642 ± 0.00199	0.0505 ± 0.00425	0.0839 ± 0.00307	0.057 ± 0.00639	0.0556 ± 0.00593
	30	0.0779 ± 0.00149	0.0634 ± 0.00605	0.104 ± 0.00345	0.0703 ± 0.00987	0.0686 ± 0.00921
	40	0.091 ± 0.0035	0.0743 ± 0.00454	0.12 ± 0.00365	0.0855 ± 0.0152	0.0831 ± 0.0141
	50	0.105 ± 0.00367	0.0849 ± 0.00349	0.134 ± 0.00333	0.0889 ± 0.00384	0.0867 ± 0.00338
CPU	10	0.0452 ± 0.00333	0.0344 ± 0.01	0.0476 ± 0.00894	0.037 ± 0.00765	0.0363 ± 0.00732
	20	0.0651 ± 0.0102	0.0588 ± 0.00834	0.0758 ± 0.00838	0.0658 ± 0.0101	0.0641 ± 0.01
	30	0.0815 ± 0.00707	0.0756 ± 0.00667	0.0937 ± 0.00842	0.0796 ± 0.0103	0.0775 ± 0.00998
	40	0.0907 ± 0.0112	0.0892 ± 0.0109	0.107 ± 0.00765	0.0926 ± 0.0108	0.0901 ± 0.0105
	50	0.101 ± 0.00963	0.0979 ± 0.00952	0.117 ± 0.00841	0.105 ± 0.0117	0.103 ± 0.0115
Boston housing	10	0.0446 ± 0.00301	0.0413 ± 0.00652	0.0635 ± 0.00478	0.0438 ± 0.00725	0.0423 ± 0.00695
	20	0.0635 ± 0.00804	0.0586 ± 0.00631	0.0906 ± 0.00548	0.0654 ± 0.0109	0.0632 ± 0.0103
	30	0.0847 ± 0.00294	0.0752 ± 0.00493	0.114 ± 0.00573	0.0779 ± 0.0073	0.0756 ± 0.00718
	40	0.0916 ± 0.00757	0.0868 ± 0.00576	0.13 ± 0.00503	0.0922 ± 0.0145	0.0894 ± 0.0142
	50	0.105 ± 0.0041	0.0976 ± 0.00375	0.144 ± 0.00468	0.1 ± 0.0159	0.0971 ± 0.0153

Table 4 continued

	%	SVT	ICkNNI	SMI	CMI	UT
Red wine	10	0.0496 ± 0.0034	0.043 ± 0.00288	0.0609 ± 0.00214	0.0433 ± 0.00269	0.0421 ± 0.00255
	20	0.0806 ± 0.0053	0.0619 ± 0.00426	0.0852 ± 0.0032	0.0614 ± 0.00366	0.0598 ± 0.00346
	30	0.0863 ± 0.0040	0.0756 ± 0.00273	0.104 ± 0.00353	0.0754 ± 0.00265	0.0733 ± 0.0025
	40	0.1167 ± 0.0046	0.0897 ± 0.00368	0.122 ± 0.00338	0.0891 ± 0.00331	0.0867 ± 0.00316
	50	0.1033 ± 0.0030	0.101 ± 0.00314	0.136 ± 0.00306	0.0979 ± 0.00252	0.0953 ± 0.00241
White wine	10	0.0472 ± 0.0029	0.0455 ± 0.00183	0.0615 ± 0.00131	0.046 ± 0.00195	0.0447 ± 0.00181
	20	0.0760 ± 0.0021	0.0653 ± 0.00202	0.0875 ± 0.00197	0.0658 ± 0.00183	0.0639 ± 0.00177
	30	0.1022 ± 0.0022	0.0811 ± 0.00153	0.107 ± 0.00187	0.0806 ± 0.00187	0.0783 ± 0.00178
	40	0.0947 ± 0.0027	0.0947 ± 0.00139	0.123 ± 0.00176	0.0931 ± 0.00117	0.0904 ± 0.00111
	50	0.1343 ± 0.0026	0.108 ± 0.00252	0.138 ± 0.00198	0.105 ± 0.00171	0.102 ± 0.00163
Diabetes	10	0.0535 ± 0.00142	0.0544 ± 0.00388	0.0581 ± 0.00311	0.0535 ± 0.00507	0.0516 ± 0.00469
	20	0.0761 ± 0.00419	0.0797 ± 0.00497	0.0851 ± 0.00437	0.0778 ± 0.00743	0.0754 ± 0.00685
	30	0.0955 ± 0.00229	0.0971 ± 0.00318	0.102 ± 0.0032	0.0934 ± 0.00401	0.0905 ± 0.00356
	40	0.108 ± 0.00622	0.113 ± 0.00398	0.118 ± 0.00385	0.108 ± 0.00502	0.104 ± 0.00465
	50	0.119 ± 0.00626	0.127 ± 0.00357	0.133 ± 0.00327	0.121 ± 0.00376	0.117 ± 0.00334
Monk1	10	0.0604 ± 0.00434	0.0658 ± 0.00653	0.067 ± 0.00959	0.0641 ± 0.00915	0.0586 ± 0.00477
	20	0.0761 ± 0.00484	0.09 ± 0.00648	0.0907 ± 0.0106	0.0872 ± 0.0102	0.08 ± 0.00653
	30	0.101 ± 0.00602	0.11 ± 0.00988	0.111 ± 0.00785	0.107 ± 0.00765	0.0997 ± 0.00615
	40	0.115 ± 0.00164	0.126 ± 0.00716	0.13 ± 0.0131	0.125 ± 0.0129	0.116 ± 0.0066
	50	0.136 ± 0.00602	0.145 ± 0.00614	0.147 ± 0.0108	0.141 ± 0.0105	0.132 ± 0.00528
Monk2	10	0.057 ± 0.00225	0.0637 ± 0.00496	0.0636 ± 0.00849	0.0612 ± 0.00816	0.0581 ± 0.00376
	20	0.0842 ± 0.00267	0.0914 ± 0.00568	0.0894 ± 0.0114	0.0861 ± 0.0109	0.0827 ± 0.00473
	30	0.103 ± 0.00493	0.11 ± 0.00585	0.111 ± 0.00985	0.107 ± 0.0094	0.101 ± 0.00382

Table 4 continued

	%	SVT	ICkNNI	SMI	CMI	UT
	40	0.118 ± 0.00577	0.126 ± 0.00617	0.128 ± 0.0116	0.123 ± 0.0114	0.117 ± 0.00421
	50	0.131 ± 0.00688	0.144 ± 0.00667	0.142 ± 0.00804	0.136 ± 0.00767	0.131 ± 0.00484
Monk3	10	0.0567 ± 0.00646	0.0628 ± 0.00726	0.0636 ± 0.00738	0.0612 ± 0.00707	0.0569 ± 0.00543
	20	0.0778 ± 0.00754	0.0909 ± 0.00684	0.0918 ± 0.0117	0.0878 ± 0.0105	0.0828 ± 0.00545
	30	0.0952 ± 0.00464	0.112 ± 0.00607	0.116 ± 0.0128	0.111 ± 0.0118	0.102 ± 0.00463
	40	0.114 ± 0.00686	0.125 ± 0.00794	0.13 ± 0.0146	0.125 ± 0.0144	0.114 ± 0.0054
	50	0.133 ± 0.00558	0.146 ± 0.00714	0.152 ± 0.0119	0.146 ± 0.0116	0.131 ± 0.00472
Avg. ranking		2.7455	2.8000	4.9455	2.9636	1.5455

Table 5 ARMSE for the QGRNN models

	%	SVT	ICkNNI	SMI	CMI	UT
Cancer	10	72.5835 ± 30.7908	61.8 ± 20.6	62.4 ± 24.1	62.8 ± 20.2	45.1 ± 9.2
	20	64.747 ± 18.9596	68 ± 29.5	64.7 ± 23.2	66.7 ± 21.6	42.3 ± 7.89
	30	58.5148 ± 13.7541	62.7 ± 17.1	63.2 ± 22.7	70.9 ± 34	42.6 ± 7.58
	40	71.651 ± 29.5705	77.2 ± 46.4	76.5 ± 42.6	69.1 ± 35.8	43.2 ± 13.1
	50	64.7058 ± 21.5201	74.7 ± 40.4	77.5 ± 48.3	63.3 ± 24.1	40.5 ± 6.82
MPG	10	3.0168 ± 0.33597	3 ± 0.326	2.97 ± 0.248	3.01 ± 0.317	2.83 ± 0.271
	20	3.0301 ± 0.42735	3.06 ± 0.388	3.08 ± 0.447	3.07 ± 0.368	2.85 ± 0.277
	30	3.0835 ± 0.35067	3.02 ± 0.356	3 ± 0.289	3 ± 0.342	2.87 ± 0.277
	40	3.1649 ± 0.34905	3.11 ± 0.314	3.08 ± 0.272	3.06 ± 0.305	2.9 ± 0.284
	50	3.1605 ± 0.3276	3.13 ± 0.446	3.16 ± 0.26	3.15 ± 0.351	2.92 ± 0.274
Compression	10	7.6515 ± 0.61114	7.73 ± 0.33	7.77 ± 0.346	7.74 ± 0.337	7.71 ± 0.313
	20	7.6856 ± 0.54843	7.82 ± 0.432	7.85 ± 0.428	7.81 ± 0.335	7.76 ± 0.322
	30	7.5971 ± 0.59638	7.84 ± 0.487	8 ± 0.405	7.89 ± 0.376	7.87 ± 0.34
	40	7.8762 ± 0.61474	7.96 ± 0.465	8.19 ± 0.513	7.92 ± 0.445	7.9 ± 0.302
	50	7.8948 ± 0.67767	7.96 ± 0.408	8.08 ± 0.35	7.94 ± 0.373	7.91 ± 0.377
CPU	10	48.7984 ± 15.9983	52.3 ± 47.7	44.1 ± 27.6	55 ± 79	87.1 ± 73.9
	20	60.7471 ± 57.7013	74.7 ± 82.4	65.8 ± 44.4	101 ± 131	79.4 ± 72.6
	30	62.8262 ± 56.9403	111 ± 113	102 ± 88.4	112 ± 107	83 ± 69.3
	40	77.4825 ± 83.4113	176 ± 410	171 ± 325	115 ± 148	82.1 ± 73.4
	50	97.3969 ± 78.4781	188 ± 189	175 ± 239	154 ± 189	70.1 ± 59.2
Boston housing	10	4.5796 ± 0.29437	4.65 ± 0.849	4.46 ± 0.849	4.58 ± 0.756	3.96 ± 0.671
	20	5.0201 ± 0.58244	4.69 ± 0.683	4.55 ± 0.809	4.66 ± 1.17	4.09 ± 0.698
	30	4.9761 ± 0.64636	4.98 ± 1.04	4.92 ± 0.993	5.39 ± 2.12	4.23 ± 0.657
	40	4.8995 ± 0.51417	5.22 ± 1.07	5.26 ± 1.21	5.24 ± 1.41	4.19 ± 0.633
	50	6.0858 ± 1.9565	5.37 ± 1.28	5.83 ± 2.31	5.21 ± 1.3	4.36 ± 0.843

Table 5 continued

	%	SVT	ICkNNI	SMI	CMI	UT
Red wine	10	0.6800 ± 0.0384	0.688 ± 0.0558	0.686 ± 0.052	0.688 ± 0.0573	0.654 ± 0.0357
	20	0.6665 ± 0.0464	0.694 ± 0.0611	0.691 ± 0.046	0.698 ± 0.0777	0.653 ± 0.0322
	30	0.6765 ± 0.0467	0.688 ± 0.0413	0.68 ± 0.0297	0.689 ± 0.0603	0.646 ± 0.0285
	40	0.6611 ± 0.0437	0.696 ± 0.036	0.697 ± 0.049	0.701 ± 0.0442	0.653 ± 0.037
	50	0.6783 ± 0.0374	0.692 ± 0.0385	0.698 ± 0.0526	0.699 ± 0.0441	0.645 ± 0.0303
White wine	10	0.7463 ± 0.1138	0.786 ± 0.135	0.787 ± 0.163	0.775 ± 0.113	0.755 ± 0.0703
	20	0.7570 ± 0.1228	0.777 ± 0.132	0.79 ± 0.151	0.791 ± 0.158	0.739 ± 0.04
	30	0.7448 ± 0.1238	0.759 ± 0.125	0.765 ± 0.0959	0.796 ± 0.188	0.739 ± 0.0301
	40	0.7836 ± 0.1467	0.803 ± 0.235	0.796 ± 0.169	0.804 ± 0.146	0.749 ± 0.0552
	50	0.7578 ± 0.1464	0.78 ± 0.143	0.775 ± 0.125	0.842 ± 0.32	0.741 ± 0.0373
Diabetes	10	0.42243 ± 0.010386	0.413 ± 0.024	0.413 ± 0.0255	0.413 ± 0.0225	0.402 ± 0.0197
	20	0.41438 ± 0.0088599	0.417 ± 0.0243	0.416 ± 0.0257	0.418 ± 0.0254	0.403 ± 0.021
	30	0.43271 ± 0.019175	0.421 ± 0.0287	0.415 ± 0.0266	0.416 ± 0.0281	0.399 ± 0.0207
	40	0.41533 ± 0.015586	0.417 ± 0.0274	0.417 ± 0.028	0.413 ± 0.027	0.4 ± 0.023
	50	0.4378 ± 0.01965	0.419 ± 0.0266	0.415 ± 0.0314	0.419 ± 0.0257	0.4 ± 0.0226
Monk 1	10	0.37539 ± 0.020015	0.385 ± 0.0198	0.384 ± 0.0174	0.387 ± 0.0186	0.389 ± 0.011
	20	0.3933 ± 0.019563	0.391 ± 0.0222	0.39 ± 0.0239	0.392 ± 0.026	0.395 ± 0.0103
	30	0.39106 ± 0.02307	0.395 ± 0.0194	0.391 ± 0.017	0.394 ± 0.0207	0.402 ± 0.0146
	40	0.39411 ± 0.031385	0.403 ± 0.0219	0.404 ± 0.0209	0.399 ± 0.0197	0.407 ± 0.0116
	50	0.40768 ± 0.019869	0.415 ± 0.0372	0.412 ± 0.0305	0.416 ± 0.0354	0.418 ± 0.0173
Monk 2	10	0.46549 ± 0.018452	0.465 ± 0.0202	0.465 ± 0.0202	0.463 ± 0.0204	0.46 ± 0.0137
	20	0.46819 ± 0.015197	0.469 ± 0.0233	0.469 ± 0.0223	0.469 ± 0.0238	0.458 ± 0.0136
	30	0.47425 ± 0.015832	0.47 ± 0.0177	0.47 ± 0.0193	0.47 ± 0.0189	0.458 ± 0.00998

Table 5 continued

	%	SVT	ICkNNI	SMI	CMI	UT
	40	0.48028 ± 0.028679	0.479 ± 0.0271	0.476 ± 0.025	0.48 ± 0.0238	0.461 ± 0.0122
	50	0.48154 ± 0.01329	0.478 ± 0.0223	0.476 ± 0.0228	0.477 ± 0.0213	0.461 ± 0.00971
Monk3	10	0.12986 ± 0.004315	0.0628 ± 0.00726	0.0636 ± 0.00738	0.0612 ± 0.00707	0.0569 ± 0.00543
	20	0.09157 ± 0.001438	0.0909 ± 0.00684	0.0918 ± 0.0117	0.0878 ± 0.0105	0.0828 ± 0.00545
	30	0.12299 ± 0.001717	0.112 ± 0.00607	0.116 ± 0.0128	0.111 ± 0.0118	0.102 ± 0.00463
	40	0.12986 ± 0.004315	0.125 ± 0.00794	0.13 ± 0.0146	0.125 ± 0.0144	0.114 ± 0.0054
	50	0.14621 ± 0.003407	0.146 ± 0.00714	0.152 ± 0.0119	0.146 ± 0.0116	0.131 ± 0.00472
Avg. ranking		2.9091	3.4000	3.4000	3.6364	1.6545

Table 6 ARMSE for the FNNRW models

	%	SVT	ICkNNI	SMI	CMI	UT
Cancer	10	71.6695 ± 9.3453	66.5 ± 7.26	69.1 ± 8.4	65.9 ± 7.79	54.4 ± 5.27
	20	70.6833 ± 9.6317	70.2 ± 12	68.7 ± 9.51	69.6 ± 9.71	48.8 ± 2.64
	30	70.6044 ± 9.7693	65 ± 7.25	65.1 ± 9.24	68.1 ± 9.36	46.7 ± 3.99
	40	69.633 ± 12.0603	64.5 ± 10.6	67.4 ± 12.4	68.7 ± 8.03	44.3 ± 3.19
	50	76.3127 ± 10.1755	70.4 ± 13	66.7 ± 10.1	69.6 ± 11	43.2 ± 3.39
MPG	10	3.5469 ± 0.45668	3.8 ± 0.49	3.65 ± 0.395	3.77 ± 0.396	2.89 ± 0.21
	20	3.879 ± 0.42399	3.98 ± 0.588	3.66 ± 0.476	4.01 ± 0.647	2.93 ± 0.176
	30	3.8407 ± 0.48589	4.12 ± 0.594	3.68 ± 0.428	3.86 ± 0.389	2.97 ± 0.215
	40	3.9695 ± 0.62831	4.35 ± 0.708	3.65 ± 0.424	3.95 ± 0.45	3.01 ± 0.234
	50	3.9961 ± 0.47673	4.24 ± 0.456	3.84 ± 0.482	4.09 ± 0.478	3.02 ± 0.212
Compression	10	8.3252 ± 0.51482	8.21 ± 0.316	8.23 ± 0.274	8.2 ± 0.225	8.05 ± 0.228
	20	8.3943 ± 0.53017	8.29 ± 0.353	8.35 ± 0.348	8.23 ± 0.316	8.21 ± 0.247
	30	8.5968 ± 0.55535	8.41 ± 0.335	8.51 ± 0.346	8.43 ± 0.312	8.35 ± 0.207
	40	8.6192 ± 0.36084	8.45 ± 0.292	8.59 ± 0.422	8.38 ± 0.237	8.45 ± 0.229
	50	8.6256 ± 0.36868	8.57 ± 0.332	8.72 ± 0.429	8.54 ± 0.314	8.5 ± 0.219
CPU	10	131.8763 ± 62.4685	123 ± 45.3	124 ± 53.8	121 ± 43.8	82.3 ± 35.8
	20	120.4793 ± 50.5714	116 ± 47.6	121 ± 46.8	114 ± 52.6	80.1 ± 34.6
	30	132.1839 ± 52.5673	107 ± 34.4	110 ± 41.8	104 ± 32.7	78.4 ± 32.6
	40	126.0762 ± 77.2748	107 ± 24.5	112 ± 41.7	115 ± 27	77 ± 33.2
	50	127.9678 ± 61.6014	116 ± 37.9	112 ± 37.4	120 ± 47.9	78.8 ± 34
Boston housing	10	4.8989 ± 0.37354	5.06 ± 0.615	5.13 ± 0.453	5.05 ± 0.584	4.63 ± 0.447
	20	5.0279 ± 0.40403	5.17 ± 0.611	5.22 ± 0.49	5.17 ± 0.559	4.64 ± 0.407
	30	5.569 ± 0.81781	5.3 ± 0.672	5.38 ± 0.599	5.23 ± 0.643	4.64 ± 0.448
	40	5.4895 ± 0.47012	5.69 ± 0.842	5.59 ± 0.577	5.39 ± 0.65	4.77 ± 0.461
	50	5.3565 ± 0.39092	5.56 ± 0.662	5.63 ± 0.478	5.38 ± 0.553	4.78 ± 0.486

Table 6 continued

	%	SVT	ICkNNI	SMI	CMI	UT
Red wine	10	0.675 ± 0.0137	0.678 ± 0.0184	0.678 ± 0.0191	0.677 ± 0.0183	0.663 ± 0.0162
	20	0.679 ± 0.0164	0.68 ± 0.0158	0.678 ± 0.0163	0.68 ± 0.0154	0.659 ± 0.0157
	30	0.681 ± 0.0189	0.682 ± 0.019	0.68 ± 0.0166	0.681 ± 0.0161	0.658 ± 0.0154
	40	0.678 ± 0.0192	0.681 ± 0.0177	0.679 ± 0.0179	0.68 ± 0.0163	0.657 ± 0.0162
	50	0.682 ± 0.0183	0.68 ± 0.0189	0.679 ± 0.015	0.681 ± 0.0182	0.657 ± 0.0167
White wine	10	0.752 ± 0.0143	0.753 ± 0.0147	0.753 ± 0.0149	0.752 ± 0.0147	0.749 ± 0.0139
	20	0.755 ± 0.0146	0.753 ± 0.0149	0.754 ± 0.0146	0.753 ± 0.0145	0.749 ± 0.0135
	30	0.754 ± 0.0153	0.753 ± 0.0145	0.755 ± 0.0139	0.753 ± 0.0144	0.75 ± 0.0132
	40	0.754 ± 0.0178	0.753 ± 0.0146	0.755 ± 0.0147	0.753 ± 0.0144	0.751 ± 0.013
	50	0.761 ± 0.0183	0.754 ± 0.0139	0.757 ± 0.0133	0.753 ± 0.0141	0.752 ± 0.0126
Diabetes	10	0.43224 ± 0.024581	0.446 ± 0.0198	0.447 ± 0.0198	0.446 ± 0.0211	0.42 ± 0.0172
	20	0.43808 ± 0.016529	0.451 ± 0.0219	0.45 ± 0.0199	0.452 ± 0.0197	0.415 ± 0.0158
	30	0.44158 ± 0.029478	0.453 ± 0.0245	0.453 ± 0.0264	0.453 ± 0.0245	0.411 ± 0.0164
	40	0.44104 ± 0.02321	0.454 ± 0.0191	0.45 ± 0.0171	0.454 ± 0.0159	0.41 ± 0.0146
	50	0.44605 ± 0.030209	0.456 ± 0.018	0.462 ± 0.0225	0.461 ± 0.0232	0.409 ± 0.0153
Monk1	10	1.4092 ± 0.18916	1.38 ± 0.349	1.44 ± 0.4	1.54 ± 0.602	0.509 ± 0.0235
	20	1.3747 ± 0.38751	1.38 ± 0.208	1.34 ± 0.253	1.45 ± 0.33	0.494 ± 0.0172
	30	1.6397 ± 0.4589	1.38 ± 0.346	1.43 ± 0.368	1.4 ± 0.351	0.487 ± 0.0171
	40	1.4804 ± 0.34607	1.39 ± 0.228	1.51 ± 0.437	1.42 ± 0.291	0.481 ± 0.0142
	50	1.6449 ± 0.29973	1.45 ± 0.3	1.44 ± 0.286	1.72 ± 0.459	0.481 ± 0.0137
Monk2	10	0.81507 ± 0.15253	0.761 ± 0.0884	0.759 ± 0.0918	0.767 ± 0.0877	0.493 ± 0.0112
	20	0.72945 ± 0.06379	0.786 ± 0.101	0.744 ± 0.0893	0.774 ± 0.119	0.49 ± 0.00946
	30	0.7535 ± 0.083207	0.801 ± 0.11	0.79 ± 0.105	0.793 ± 0.11	0.485 ± 0.0084

Table 6 continued

	%	SVT	ICkNNI	SMI	CMI	UT
	40	0.78518 ± 0.071291	0.8 ± 0.0958	0.779 ± 0.065	0.805 ± 0.0873	0.487 ± 0.0116
	50	0.78789 ± 0.066104	0.808 ± 0.0795	0.795 ± 0.0714	0.838 ± 0.0978	0.488 ± 0.00898
Monk3	10	1.0002 ± 0.17404	1.16 ± 0.272	1.14 ± 0.289	1.12 ± 0.311	0.411 ± 0.0152
	20	1.1875 ± 0.26235	1.24 ± 0.34	1.24 ± 0.305	1.19 ± 0.234	0.396 ± 0.0124
	30	1.2316 ± 0.34527	1.2 ± 0.194	1.14 ± 0.243	1.22 ± 0.278	0.394 ± 0.0138
	40	1.3447 ± 0.25644	1.22 ± 0.272	1.24 ± 0.28	1.25 ± 0.279	0.393 ± 0.014
	50	1.5019 ± 0.29112	1.32 ± 0.433	1.35 ± 0.306	1.38 ± 0.425	0.392 ± 0.0136
Avg. ranking		3.6182	3.3818	3.4727	3.4909	1.0364

Table 7 Comparison of different approaches to estimate the Q-Gaussian function in presence of missing entries on real-world data

	%	SVT	ICkNNI	SMI	CMI	UT
Cancer	10	0.0184 ± 0.0241	0.0459 ± 0.00522	0.0693 ± 0.00811	0.037 ± 0.00876	0.0363 ± 0.00859
	20	0.0282 ± 0.0239	0.0679 ± 0.00624	0.099 ± 0.00744	0.0542 ± 0.00916	0.053 ± 0.00893
	30	0.0402 ± 0.0299	0.088 ± 0.00773	0.124 ± 0.0082	0.0665 ± 0.00945	0.065 ± 0.00939
	40	0.034 ± 0.0123	0.103 ± 0.00841	0.141 ± 0.00955	0.0749 ± 0.00956	0.0733 ± 0.00944
	50	0.0399 ± 0.0181	0.115 ± 0.00736	0.157 ± 0.00758	0.0852 ± 0.0126	0.0834 ± 0.0124
MPG	10	0.0407 ± 0.0106	0.0394 ± 0.00483	0.063 ± 0.00505	0.0387 ± 0.00421	0.0377 ± 0.00396
	20	0.0685 ± 0.018	0.0528 ± 0.00662	0.088 ± 0.00487	0.0549 ± 0.0107	0.0536 ± 0.0102
	30	0.0859 ± 0.0148	0.0661 ± 0.00608	0.109 ± 0.00441	0.0685 ± 0.014	0.0667 ± 0.0128
	40	0.094 ± 0.0139	0.0796 ± 0.00729	0.126 ± 0.0067	0.0798 ± 0.0104	0.0778 ± 0.00977
	50	0.101 ± 0.0142	0.0876 ± 0.00699	0.14 ± 0.00551	0.0894 ± 0.012	0.0872 ± 0.0115
Compression	10	0.0388 ± 0.00706	0.0348 ± 0.00508	0.0598 ± 0.00368	0.0403 ± 0.00511	0.0393 ± 0.00481
	20	0.0597 ± 0.00465	0.0505 ± 0.00425	0.0839 ± 0.00307	0.057 ± 0.00639	0.0556 ± 0.00593
	30	0.0673 ± 0.00679	0.0634 ± 0.00605	0.104 ± 0.00345	0.0703 ± 0.00987	0.0686 ± 0.00921
	40	0.073 ± 0.00588	0.0743 ± 0.00454	0.12 ± 0.00365	0.0855 ± 0.0152	0.0831 ± 0.0141
	50	0.088 ± 0.00483	0.0849 ± 0.00349	0.134 ± 0.00333	0.0889 ± 0.00384	0.0867 ± 0.00338
CPU	10	0.0918 ± 0.0379	0.0344 ± 0.01	0.0476 ± 0.00894	0.037 ± 0.00765	0.0363 ± 0.00732
	20	0.281 ± 0.303	0.0588 ± 0.00834	0.0758 ± 0.00838	0.0658 ± 0.0101	0.0641 ± 0.01
	30	0.227 ± 0.136	0.0756 ± 0.00667	0.0937 ± 0.00842	0.0796 ± 0.0103	0.0775 ± 0.00998
	40	0.337 ± 0.29	0.0892 ± 0.0109	0.107 ± 0.00765	0.0926 ± 0.0108	0.0901 ± 0.0105
	50	0.633 ± 0.441	0.0979 ± 0.00952	0.117 ± 0.00841	0.105 ± 0.0117	0.103 ± 0.0115
Boston housing	10	0.0301 ± 0.00806	0.0413 ± 0.00652	0.0635 ± 0.00478	0.0438 ± 0.00725	0.0423 ± 0.00695
	20	0.0603 ± 0.0127	0.0586 ± 0.00631	0.0906 ± 0.00548	0.0654 ± 0.0109	0.0632 ± 0.0103
	30	0.0575 ± 0.0153	0.0752 ± 0.00493	0.114 ± 0.00573	0.0779 ± 0.0073	0.0756 ± 0.00718
	40	0.103 ± 0.0651	0.0868 ± 0.00576	0.13 ± 0.00503	0.0922 ± 0.0145	0.0894 ± 0.0142
	50	0.102 ± 0.0308	0.0976 ± 0.00375	0.144 ± 0.00468	0.1 ± 0.0159	0.0971 ± 0.0153

Table 7 continued

	%	SVT	ICkNNI	SMI	CMI	UT
Red wine	10	0.044 ± 0.00274	0.043 ± 0.00288	0.0609 ± 0.00214	0.0433 ± 0.00269	0.0421 ± 0.00255
	20	0.0673 ± 0.00386	0.0619 ± 0.00426	0.0852 ± 0.0032	0.0614 ± 0.00366	0.0598 ± 0.00346
	30	0.0764 ± 0.00343	0.0756 ± 0.00273	0.104 ± 0.00353	0.0754 ± 0.00265	0.0733 ± 0.0025
	40	0.0834 ± 0.00412	0.0897 ± 0.00368	0.122 ± 0.00338	0.0891 ± 0.00331	0.0867 ± 0.00316
	50	0.0974 ± 0.00319	0.101 ± 0.00314	0.136 ± 0.00306	0.0979 ± 0.00252	0.0953 ± 0.00241
White wine	10	0.0488 ± 0.0064	0.0455 ± 0.00183	0.0615 ± 0.00131	0.046 ± 0.00195	0.0447 ± 0.00181
	20	0.0810 ± 0.0061	0.0653 ± 0.00202	0.0875 ± 0.00197	0.0658 ± 0.00183	0.0639 ± 0.00177
	30	0.0798 ± 0.0065	0.0811 ± 0.00153	0.107 ± 0.00187	0.0806 ± 0.00187	0.0783 ± 0.00178
	40	0.0910 ± 0.0073	0.0947 ± 0.00139	0.123 ± 0.00176	0.0931 ± 0.00117	0.0904 ± 0.00111
	50	0.1035 ± 0.0069	0.108 ± 0.00252	0.138 ± 0.00198	0.105 ± 0.00171	0.102 ± 0.00163
Diabetes	10	0.0537 ± 0.0124	0.0551 ± 0.0127	0.0539 ± 0.015	0.0188 ± 0.0188	0.0517 ± 0.0197
	20	0.0788 ± 0.0107	0.096 ± 0.0434	0.0915 ± 0.0429	0.043 ± 0.043	0.0887 ± 0.043
	30	0.0877 ± 0.00836	0.116 ± 0.0364	0.109 ± 0.0347	0.0364 ± 0.0364	0.105 ± 0.0361
	40	0.121 ± 0.0257	0.135 ± 0.0406	0.137 ± 0.038	0.052 ± 0.052	0.135 ± 0.0631
	50	0.144 ± 0.0367	0.138 ± 0.0342	0.139 ± 0.033	0.0402 ± 0.0402	0.135 ± 0.0431
Monk1	10	0.0395 ± 0.00196	0.0383 ± 0.00449	0.0418 ± 0.0041	0.00507 ± 0.00507	0.0376 ± 0.00437
	20	0.061 ± 0.00322	0.0547 ± 0.00527	0.0584 ± 0.00355	0.00702 ± 0.00702	0.052 ± 0.00683
	30	0.0719 ± 0.0052	0.0683 ± 0.00406	0.0852 ± 0.00399	0.0546 ± 0.0546	0.0799 ± 0.0666
	40	0.0796 ± 0.00195	0.0794 ± 0.00371	0.0867 ± 0.00269	0.00679 ± 0.00679	0.078 ± 0.00713
	50	0.0902 ± 0.00276	0.0883 ± 0.00432	0.106 ± 0.00363	0.0423 ± 0.0423	0.101 ± 0.0616
Monk2	10	0.0391 ± 0.00148	0.039 ± 0.00376	0.0428 ± 0.00339	0.00534 ± 0.00534	0.0386 ± 0.00505
	20	0.056 ± 0.00325	0.0562 ± 0.00339	0.0586 ± 0.00334	0.00457 ± 0.00457	0.0528 ± 0.00568
	30	0.0691 ± 0.00414	0.0692 ± 0.00395	0.0748 ± 0.00286	0.00688 ± 0.00688	0.067 ± 0.00881

Table 7 continued

	%	SVT	ICkNNI	SMI	CMI	UT
	40	0.0793 ± 0.00463	0.0791 ± 0.00394	0.0921 ± 0.00287	0.0337 ± 0.0337	0.0868 ± 0.052
	50	0.0895 ± 0.00273	0.0901 ± 0.00459	0.0981 ± 0.00418	0.00843 ± 0.00843	0.0882 ± 0.00963
Monk3	10	0.0376 ± 0.00217	0.04 ± 0.0051	0.0423 ± 0.00439	0.00545 ± 0.00545	0.0383 ± 0.00556
	20	0.0599 ± 0.004	0.0552 ± 0.00493	0.0592 ± 0.004	0.00512 ± 0.00512	0.0531 ± 0.00476
	30	0.0666 ± 0.00292	0.0706 ± 0.00551	0.0761 ± 0.0048	0.00926 ± 0.00926	0.0676 ± 0.00979
	40	0.0781 ± 0.00226	0.0813 ± 0.006	0.0885 ± 0.00475	0.0098 ± 0.0098	0.0794 ± 0.0107
	50	0.0919 ± 0.0037	0.0896 ± 0.00484	0.102 ± 0.00412	0.0372 ± 0.0372	0.0984 ± 0.0647
Avg. ranking		3.0182	2.6909	4.8000	2.3455	2.1455

shall be highlighted it that our model has an automatically selected regularization term. In Eq. (9), we can see matrix P as a regularization term that depends on the uncertainty of the estimates. Thus, our formulation can be seen as an uncertainty robust approach.

Once again, the statistical significance of our results was tested with the Friedman test. The null hypothesis H_0 was rejected for the FNNRW with $p = 2.3275e - 17$ and with $p = 5.9173e - 08$ for the QGRNN. The $CD = 0.82251$ and the average rankings showed that the models based on our proposals significantly outperformed all other approaches.

6 Conclusion

In this paper, we presented a methodology to design NNRWs for datasets with missing components. Our proposal presents a training procedure for NNRWs which can profit from incomplete feature vectors without the need of a previous imputation step.

In the proposed approach, data is modeled with a GMM, and the output of the hidden layer is estimated in terms of its expected values and variances. Such statistical measures are used to solve a robust least-squares problem to estimate the coefficients of the output layer.

To validate our method, we perform three sets of experiments. The first two sets aim to verify how well our method estimates basis functions that are part of a NNRW. We considered the logistic and the q-Gaussian basis functions as case studies. Our experiments included simulated and real datasets. The last experiment tested our proposal on two NNRW, the QGRNN and the FNNRW, in real datasets.

The results obtained show that our proposal was able to estimate the basis function more accurately and also result in NNRW with better performances for almost all tested cases. It is worth noting that our method was compared to both parametric and non-parametric approaches. A statistical analysis of the results shows that the superior performance of our method is significant. Even though the method showed promising results, it is important to point that its performance is strongly related to the chosen distribution estimation procedure (in this paper, we used a GMM). Another import aspect that shall be highlighted is that the UT consists of an approximated procedure to obtain the moments of the distribution, thus exact algorithm shall be preferred when available (see [32,34,38] for some examples). In future works, we intend to compare our proposal to other recent imputation methods and also analyze the impact of our strategy in other models like RVFL and deep neural networks.

Acknowledgements The authors would like to thank the Brazilian National Council for Scientific and Technological Development (CNPq) for the financial support (Grant No. 305048/2016-3)

Appendix: Unscented Transform (UT)

Given a D-dimensional random variable X, we are interested in estimating statistical moments of ψ, which results from an application of a non-linear function $h(\cdot)$ to X. These values could be obtained via standard sampling procedures or numerical integration methods. However, such procedures can be computationally intensive and depend on many factors such as proper initialization, stop criteria, etc. The Unscented Transform (UT) provides a scheme to estimate the moments of ψ using a small set of deterministically chosen samples, referred to as sigma points (SPs), from the space of X.

There are different possible ways to choose the SPs. A common approach is to use a symmetric set of $S = 2D + 1$ SPs as described in Eqs. (32) to (34).

$$\gamma_1 = \mathbb{E}[X] \tag{32}$$

$$\gamma_s = \gamma_1 + \left[\sqrt{D\,\Sigma}\right]_{s-1} \qquad \forall\, 1 < s \leq D+1 \tag{33}$$

$$\gamma_s = \gamma_1 - \left[\sqrt{D\,\Sigma}\right]_{s-(D+1)} \qquad \forall\, D+1 < s \leq 2D+1 \tag{34}$$

where $\left[\sqrt{D\,\Sigma}\right]_s$ denotes the s-th row of the matrix square root of $D\,\Sigma$, which is the covariance matrix Σ of X.

Given the SPs and a set of weights $\{k_s\}_{s=1}^{S} \subset \mathbb{R}$, we can approximate the moments of ψ using a simple set of rules. For instance $\mathbb{E}[\psi]$ and $\mathrm{Cov}(\psi)$ can then be approximated using the following equations:

$$\delta_s \leftarrow h\left(\gamma_s\right) \quad \forall\, 1 \leq s \leq S, \tag{35}$$

$$\mathbb{E}[\psi] \approx \sum_{s=1}^{S} k_s \delta_s \tag{36}$$

$$\mathrm{Cov}(\psi) \approx \sum_{s=1}^{S} k_s \left(\delta_s - \mathbb{E}[\psi]\right)\left(\delta_s - \mathbb{E}[\psi]\right)^T. \tag{37}$$

Although there is no restriction on their sign, the weights $k_1, \ldots, k_S$ must respect the convexity constraint.

$$\sum_{s=1}^{S} k_s = 1, \tag{38}$$

to provide an unbiased estimate [22]. In this paper, we set $k_1 = k_2 = \cdots = k_S = 1/S$.

References

1. Abdella M, Marwala T (2005) The use of genetic algorithms and neural networks to approximate missing data in database. In: IEEE 3rd international conference on computational cybernetics ICCC 2005, pp 207–212
2. Braake HAT, Straten GV (1995) Random activation weight neural net (rawn) for fast non-iterative training. Eng Appl Artif Intell 8(1):71–80. https://doi.org/10.1016/0952-1976(94)00056-S
3. Broomhead DS, Lowe D (1988) Multivariable functional interpolation and adaptive networks. Complex Syst 2:321–355
4. Cai J, Candès E, Shen Z (2010) A singular value thresholding algorithm for matrix completion. SIAM J Optim 20(4):1956–1982. https://doi.org/10.1137/080738970
5. Cox D, Pinto N (2011) Beyond simple features: a large-scale feature search approach to unconstrained face recognition. Face Gesture 2011:8–15. https://doi.org/10.1109/FG.2011.5771385
6. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control, Signals Syst 2(4):303–314
7. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
8. Ding Y, Simonoff JS (2010) An investigation of missing data methods for classification trees applied to binary response data. J Mach Learn Res 11:131–170
9. Eirola E, Lendasse A, Vandewalle V, Biernacki C (2014) Mixture of gaussians for distance estimation with missing data. Neurocomputing 131:32–42
10. Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11(1):86–92
11. Funahashi KI (1989) On the approximate realization of continuous mappings by neural networks. Neural Netw 2(3):183–192

12. Garcia-Laencina PJ, Sancho-Gomez JL, Figueiras-Vidal AR (2010) Pattern classification with missing data: a review. Neural Comput Appl 19(2):263–282
13. Giryes R, Sapiro G, Bronstein AM (2016) Deep neural networks with random gaussian weights: a universal classification strategy? IEEE Trans Signal Process 64:3444–3457
14. Guo P (2018) A vest of the pseudoinverse learning algorithm. CoRR arXiv:1805.07828
15. Guo P, Chen PC, Sun Y (1995) An exact supervised learning for a three-layer supervised neural network. In: International conference on neural information processing (ICONIP), Beijing, pp 1041–1044
16. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
17. Hulse JV, Khoshgoftaar TM (2014) Incomplete-case nearest neighbor imputation in software measurement data. Inf Sci 259:596–610
18. Hunt L, Jorgensen M (2003) Mixture model clustering for mixed data with missing information. Comput Stat Data Anal 41(3–4):429–440
19. Gheyas IA, Smith LS (2010) A neural network-based framework for the reconstruction of incomplete data sets. Neurocomputing 73(16–18):3039–3065
20. Igelnik B, Pao YH (1995) Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans Neural Netw 6(6):1320–1329. https://doi.org/10.1109/72.471375
21. Julier SJ, Uhlmann JK (1997) A new extension of the Kalman filter to nonlinear systems. In: SPIE aerosense symposium, pp 182–193
22. Julier SJ, Uhlmann JK (2004) Unscented filtering and nonlinear estimation. Proc IEEE 92(3):401–422
23. Kang P (2013) Locally linear reconstruction based missing value imputation for supervised learning. Neurocomputing 118:65–78
24. Leão BP, Yoneyama T (2011) On the use of the unscented transform for failure prognostics. In: IEEE aerospace conference. IEEE, Big Sky
25. Li C, Zhou H (2017) svt: Singular value thresholding in MATLAB. J Stat Softw, Code Snippets 81(2):1–13. https://doi.org/10.18637/jss.v081.c02
26. Li M, Wang D (2017) Insights into randomized algorithms for neural networks: practical issues and common pitfalls. Inf Sci 382–383:170–178. https://doi.org/10.1016/j.ins.2016.12.007
27. Li Y, Yu W (2017) A fast implementation of singular value thresholding algorithm using recycling rank revealing randomized singular value decomposition. CoRR arXiv:1704.05528
28. Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 5 Jan 2018
29. Little RJA, Rubin DB (2002) Statistical analysis with missing data. Wiley, Hoboken
30. Luengo J, García S, Herrera F (2010) A study on the use of imputation methods for experimentation with radial basis function network classifiers handling missing attribute values: the good synergy between RBFNs and eventcovering method. Neural Netw 23(3):406–418
31. Meng XL, Rubin DB (1993) Maximum likelihood estimation via the ecm algorithm: a general framework. Biometrika 80(2):267–278
32. Mesquita DP, Gomes JP, Souza AH Jr, Nobre JS (2017) Euclidean distance estimation in incomplete datasets. Neurocomputing 248:11–18. https://doi.org/10.1016/j.neucom.2016.12.081
33. Mesquita DP, Gomes JP, Corona F, Souza AH, Nobre JS (2019) Gaussian kernels for incomplete data. Appl Soft Comput 77:356–365. https://doi.org/10.1016/j.asoc.2019.01.022
34. Mesquita DPP, Gomes JPP, Souza AH Jr (2017) Epanechnikov kernel for incomplete data. Electron Lett 53(21):1408–1410. https://doi.org/10.1049/el.2017.0507
35. Oliveira PG, Coelho AL (2009) Genetic versus nearest-neighbor imputation of missing attribute values for RBF networks. In: Koppen M, Kasabov N, Coghill G (eds) Advances in neuro-information processing. Springer, Berlin, pp 276–283
36. Pao YH, Phillips SM, Sobajic DJ (1992) Neural-net computing and the intelligent control of systems. Int J Control 56(2):263–289
37. Pao YH, Park GH, Sobajic DJ (1994) Learning and generalization characteristics of the random vector functional-link net. Neurocomputing 6(2):163–180. https://doi.org/10.1016/0925-2312(94)90053-1
38. Pelckmans K, Brabanter JD, Suykens J, Moor BD (2005) Handling missing values in support vector machine classifiers. Neural Netw 18(5–6):684–692
39. Pinto N, Doukhan D, DiCarlo JJ, Cox DD (2009) A high-throughput screening approach to discovering good forms of biologically inspired visual representation. PLOS Comput Biol 5(11):1–12. https://doi.org/10.1371/journal.pcbi.1000579
40. Rudi A, Rosasco L (2017) Generalization properties of learning with random features. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems, vol 30. Curran Associates, Inc., pp 3215–3225. http://papers.nips.cc/paper/6914-generalization-properties-of-learning-with-random-features.pdf

41. Saxe AM, Koh PW, Chen Z, Bhand M, Suresh B, Ng AY (2011) On random weights and unsupervised feature learning. In: Proceedings of the 28th international conference on machine learning ICML'11. Omnipress, Madison, pp 1089–1096
42. Scardapane S, Wang D (2017) Randomness in neural networks: an overview. Wiley Interdisc Rev: Data Min Knowl Discov 7:e1200
43. Schmidt WF, Kraaijveld MA, Duin RPW (1992) Feedforward neural networks with random weights. In: Proceedings, 11th IAPR international conference on pattern recognition, conference B: pattern recognition methodology and systems, vol 2, pp 1–4
44. Smola AJ, Vishwanathan SVN, Hofmann T (2005) Kernel methods for missing variables. In: Proceedings of the tenth international workshop on artificial intelligence and statistics, pp 325–332
45. Stosica D, Stosic D, Zanchettin C, Ludermir T, Stosic B (2017) QRNN: q-generalized random neural network. IEEE Trans Neural Netw Learn Syst 28(2):383–390
46. Suganthan PN (2018) Letter: on non-iterative learning algorithms with closed-form solution. Appl Soft Comput 70:1078–1082. https://doi.org/10.1016/j.asoc.2018.07.013
47. Vidya L, Vivekanand V, Shyamkumar U, Mishra D (2015) RBF-network based sparse signal recovery algorithm for compressed sensing reconstruction. Neural Netw 63:66–78
48. Wang D, Li M (2017) Deep stochastic configuration networks: universal approximation and learning representation. CoRR arXiv:1702.05639
49. Wang D, Li M (2017) Stochastic configuration networks: fundamentals and algorithms. IEEE Trans Cyber 47(10):3466–3479. https://doi.org/10.1109/TCYB.2017.2734043
50. Yu Q, Miche Y, Eirola E, van Heeswijk M, SÃl'verin E, Lendasse A (2013) Regularized extreme learning machine for regression with missing data. Neurocomputing 102:45–51
51. Ding Z, Fu Y (2018) Deep domain generalization with structured low-rank constraint. IEEE Trans Image Process 27(1):304–313. https://doi.org/10.1109/TIP.2017.2758199
52. Zhang L, Suganthan P (2016) A survey of randomized algorithms for training neural networks. Inf Sci 364–365:146–155. https://doi.org/10.1016/j.ins.2016.01.039