HANDLING MISSING VALUES VIA A NEURAL SELECTIVE INPUT MODEL

Noel Lopes[*], *Bernardete Ribeiro*[†]

Abstract: Missing data represent an ubiquitous problem with numerous and diverse causes. Handling Missing Values (MVs) properly is a crucial issue, in particular in Machine Learning (ML) and pattern recognition. To date, the only option available for standard Neural Networks (NNs) to handle this problem has been to rely on pre-processing techniques such as imputation for estimating the missing data values, which limited considerably the scope of their application. To circumvent this limitation we propose a Neural Selective Input Model (NSIM) that accommodates different transparent and bound models, while providing support for NNs to handle MVs directly. By embedding the mechanisms to support MVs we can obtain better models that reflect the uncertainty caused by unknown values. Experiments on several UCI datasets with both different distributions and proportion of MVs show that the NSIM approach is very robust and yields good to excellent results. Furthermore, the NSIM performs better than the state-of-the-art imputation techniques either with higher prevalence of MVs in a large number of features or with a significant proportion of MVs, while delivering competitive performance in the remaining cases. We demonstrate the usefulness and validity of the NSIM, making this a first-class method for dealing with this problem.

Key words: *Missing Values, Neural Networks, Back-Propagation, Multiple Back-Propagation*

Received: April 17, 2012
Revised and accepted: July 10, 2012

1. Introduction

Incomplete data pose an unavoidable problem for most real-world databases which often contain missing data [1, 2]. In particular, in domains such as gene expression

[*]Noel Lopes
UDI/IPG – Research Unit, Polytechnic Institute of Guarda, Portugal; CISUC – Center for Informatics and Systems of University of Coimbra, Portugal, E-mail: noel@ipg.pt
[†]Bernardete Ribeiro
Department of Informatics Engineering, University of Coimbra, Portugal; CISUC – Center for Informatics and Systems of University of Coimbra, Portugal, E-mail: bribeiro@dei.uc.pt

microarray experiments or clinical medicine, databases routinely lack pieces of information [3, 4]. Missing Values (MVs) can exist either by design (e.g. a survey questionnaire may allow people to leave unanswered questions) or by a combination of several other factors which prevent the data from being collected and/or stored.

The reasons for the prevalence of MVs include, among others, sensors failure, malfunction of the equipment used to record the data, data transmission problems, different patients performing different medical diagnosis tests according to their doctor and insurance coverage, merging two or more databases with a different set of attributes [5, 6, 7].

Independently of the causes associated to the existence of MVs, the fact is that most scientific data procedures are not designed to handle them [8]. In particular in the Machine Learning (ML) area, many of the most prominent algorithms (e.g. support vector machines, Neural Networks (NNs)) fail to consider MVs at all. Nevertheless, handling them in a proper manner has become a fundamental requirement for building accurate models and failure to do so usually results in models with large errors [5].

To circumvent the Missing Values Problem (MVP), ML algorithms usually rely on data pre-processing techniques such as imputation for estimating the missing data. Hence, in this case, estimated data will have the same relevance and credibility of real-data. Thus, wrong estimates of crucial variables can substantially weaken the capacity of generalization of the resulting model and originate unpredicted and potentially dramatic results [9]. Moreover, estimation methods such as imputation were conventionally developed and validated under the assumption that MVs occur in a random manner. Nevertheless this assumption does not always hold in practice [3].

In this paper we present a solution for the MVP that does not rely on estimation methods or any other data pre-processing technique and takes into consideration the uncertainty produced by the absence of such values. The proposed Neural Selective Input Model (NSIM) accounts for the creation of different transparent NN models according to the missing features. The resulting models are bound to share information among them. This paper extends the work proposed in Lopes and Ribeiro [10] by presenting a formal approach and conducting additional (and more exhaustive) tests as well as a statistical analysis to demonstrate the validity of the proposed strategy as compared to traditional methods.

The remainder of the paper is structured as follows. In the next section we describe several techniques to address the MVP. In Section 3 we describe the proposed method. Section 4 presents and discusses the results obtained for well-known benchmarks with different distributions and proportion of MVs. Finally, in Section 5 conclusions and future work are addressed.

2. Missing Values Related Work

2.1 Missing data mechanisms

The presence of MVs in data observations is one of the most frequent problems that researchers face when building ML systems [11]. Given an input matrix $\boldsymbol{\chi} \in \mathbb{R}^{n \times d}$, containing n samples and d features, we can build a binary response indicator

matrix, $\boldsymbol{\tau} \in \{0,1\}^{n \times d}$ such that:

$$\tau_{ij} = \begin{cases} 1 & \text{if } \chi_{ij} \text{ is observed} \\ 0 & \text{if } \chi_{ij} \text{ is missing} \end{cases} . \tag{1}$$

Assuming we divide $\boldsymbol{\chi}$ into the observed input dataset, $\boldsymbol{\chi}_{obs}$, containing the variables (features) for which all values are known, and into the unknown input set, $\boldsymbol{\chi}_{miss}$ containing the variables with MVs:

$$\boldsymbol{\chi} = \{\boldsymbol{\chi}_{obs}, \boldsymbol{\chi}_{miss}\} \tag{2}$$

we can define the conditional distribution for the missing data as:

$$p(\boldsymbol{\tau} \mid \boldsymbol{\chi}, \xi) = p(\boldsymbol{\tau} \mid \boldsymbol{\chi}_{obs}, \boldsymbol{\chi}_{miss}, \xi), \tag{3}$$

where ξ denotes the unknown parameters which define the missing data mechanism [5, 12]. Little and Rubin [13] define three types of missing data mechanisms according to their causes: Missing At Random (MAR), Missing Completely At Random (MCAR) and Not Missing At Random (NMAR).

2.1.1 Missing At Random (MAR)

The data are said to be MAR if the causes for their absence are independent of the missing variables, but traceable or predictable from other observed variables [5]. In such a case we can define the conditional distribution for the missing data as (4):

$$p(\boldsymbol{\tau} \mid \boldsymbol{\chi}_{obs}, \boldsymbol{\chi}_{miss}, \xi) = p(\boldsymbol{\tau} \mid \boldsymbol{\chi}_{obs}, \xi). \tag{4}$$

Examples of data MAR occur in the following cases:

- While answering questions in a survey, project managers may skip those related to small projects more often than those related to larger projects, because they may remember less details about smaller projects [12].
- A sensor occasionally fails due to power outages, preventing the data acquisition process from taking place [5].

In both cases, the cause for the absence is not directly tied to the variables containing the MVs but rather to other external influences. In the first case the MAR assumption can apply, because the predictor "project size" explains the likelihood of the value to be missing [12]. Similarly, the power outages in the second case explain why the sensor data are missing.

2.1.2 Missing Completely At Random (MCAR)

Data are said to be MCAR when the probability that a given variable is missing is independent of the variable itself and of any other external influences of interest, i.e. the reason for the MVs is completely random. This condition can be expressed as (5):

$$p(\boldsymbol{\tau} \mid \boldsymbol{\chi}_{obs}, \boldsymbol{\chi}_{miss}, \xi) = p(\boldsymbol{\tau} \mid \xi). \tag{5}$$

Examples of this mechanism include the following [5]:

- A biological sample is accidentally contaminated by the researcher collecting the data.
- A page from a questionnaire is unintentionally lost.

2.1.3 Not Missing At Random (NMAR)

The alternative for data MAR or MCAR is to consider that the data are NMAR. This is the case when the pattern of data absence depends on the missing variables themselves. A typical example of data NMAR is in the case of a personal survey involving private questions where the nature of some questions will most likely leave them unanswered. In this scenario, unless the survey can reliably measure variables that are strongly related to those containing MVs, the MAR and MCAR assumptions are violated and we must consider that data are NMAR [12].

When data are NMAR valuable information is lost and there is no general method for handling MVs properly. Otherwise, the missing data mechanism is termed ignorable and the cause for missing data can simply be ignored, allowing researchers to simplify the methods used for handling MVs [5].

2.2 Methods for Handling MVs in Machine Learning

According to Laencina et al. [5], there are four types of methods to handle MVs: case deletion, missing data imputation, model-based procedures and ML methods for handling missing data. Our view is that those can be further classified into pre-processing techniques and algorithms with built-in support for MVs. Fig. 1 presents an overview of ML procedures to handle MVs.

Since many algorithms cannot directly handle MVs, a common practice is to rely on data pre-processing techniques. Usually, this is accomplished by using imputation or simply by removing instances (case deletion) and/or features containing

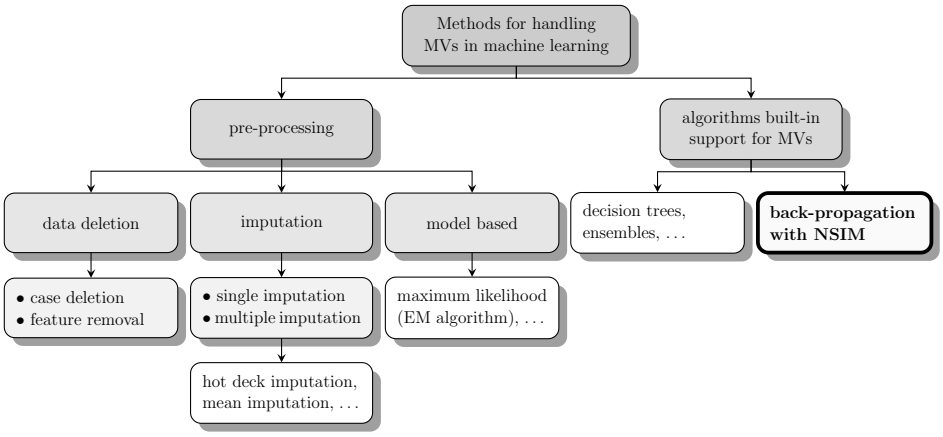

Fig. 1 *Overview of the types of techniques for handling MVs in ML.*

MVs [5, 3, 11, 12, 17, 18, 1]. A review of the methods and techniques to deal with this problem, including a comparison of some well-known approaches, can be found in Laencina et al. [5].

Removing features or instances containing a large fraction of MVs is a common (and appealing) approach for dealing with the MVP, because it is a simple process and reduces the dimensionality of the data (therefore potentially reducing the complexity of the problem). This is a very conservative strategy which guarantees that errors are not introduced in the dataset [6]. However, it is not applicable when the MVs cover a large fraction of the instances, or when their presence in essential attributes is large [18, 6]. In some cases, MVs represent only a small fraction of the data but they spread throughout a large number of instances, rendering the option of case deletion inviable. Such scenario usually happens for datasets containing a large number of features with MVs [11]. Moreover, for some problems the number of samples available is reduced and removing instances with MVs is simply not affordable. Furthermore, discarding data may damage the reliability of the derived models [6]: if the eliminated instances are dissimilar to the remaining ones, the resulting models may not be able to properly capture the underlying data distribution and consequently will suffer from bad generalization performance [11]. Likewise, by removing features it is assumed that their information is either irrelevant or it can be compensated by other variables. However, this is not always the case and features containing MVs may have critical information which cannot be compensated by the information embedded in the remaining features.

An alternative for deleting data containing MVs consists of estimating their values. Naturally, this process can introduce noise into the dataset and if a variable value is not meaningful for a given set of instances any attempt to substitute the MVs by an estimate is likely to lead to invalid results [6]. Many algorithms have been developed for this purpose (e.g. mean imputation, regression imputation, hot deck imputation, weighted k-nearest neighbor approach, Bayesian principle component analysis, local least squares) [5, 3, 17, 13]. In the NN domain, an example of such an approach consists of using a Hopfield network, in which neurons are considered both inputs and outputs, as an autoassociative memory [14, 15]. Basically, when the Hopfield network receives a noisy or incomplete pattern, it will iterate to a stable state that best matches the unknown input pattern [15, 16]. Independently of the chosen method, wrong estimates of crucial variables can substantially weaken the capacity of generalization of the resulting models and originate unpredicted and potentially dramatic outcomes. Moreover, models created using imputed (estimated) data consider MVs as if they are the real ones (albeit their value is not known) and, therefore, the resulting conclusions do not show the uncertainty produced by the absence of such values [11]. Furthermore, statistically, the variability or correlation estimations can be strongly biased [11].

Multiple imputation techniques (e.g. metric matching, Bayesian bootstrap) take into account the variability produced by the absence of MVs, by replacing each MV with two or more acceptable values, representing a distribution of possibilities [11]. However, although the variability is taken into account, MVs will still be treated as if they are real. Furthermore, estimation methods were conventionally developed and validated under the assumption that data are MAR. However, this assumption does not always hold in practice. In the particular case of microarray

experiments the distribution of missing data is highly non-random due to technical and/or experimental conditions [3].

Pre-processing methods have the advantage of allowing the same data to be used by different algorithms. Nevertheless, the burden of pre-processing data, which already accounts for most of the time that is spent to build an ML system, is further increased. Moreover, additional knowledge is required to provide a better foundation for making decisions on choosing strategic options, namely, methods and tools available to handle MVs.

Finally, these methods result in the loss of information when the absence is by itself informative. This is the case when the MVs distribution provides valuable information for the classification task that is lost when the MVs are replaced by their respective estimates [5]. For example in a bankruptcy real-world problem, distressed companies tend to omit much more financial information than healthy ones [9]. A simple explanation for this behavior is that, in general, companies in the process of insolvency try to conceal their true financial situation from its stakeholders (suppliers, customers, employees, creditors, investors, etc.). Thus, in this particular case, the specialized knowledge that a particular set of data variables is missing can play an important role in the construction of a better model.

Algorithms with built-in support for handling MVs offer numerous advantages over (more) conventional ones: (i) the amount of time spent in the pre-processing phase is reduced; (ii) researchers and practitioners are relieved from this expensive task; (iii) their performance is consistent and does not depend on the knowledge of proper methods and tools for handling MVs (e.g. some practitioners may rely on ad-hoc solutions and obtain less reliable models than those built with better skilled techniques); (iv) noise and outliers are not inadvertently injected in the data and the uncertainty associated to the missing data is preserved; (v) it can be decided whether the missing information is informative (or not) resulting in better models. Despite these advantages, most algorithms are incapable of handling MVs, mostly because in many cases that would complicate their methods to the point that they could become impractical and in many situations there would not be any real gains. Fortunately, this is not always the case and in this paper we present an elegant and simple solution that empowers NNs with the ability of dealing directly with the ubiquitous MVP.

3. Proposed Approach

The building blocks of the proposed NSIM are the selective activation neurons of a multi-layer neural network whose importance and contribution to the NN output depends on the stimulus presented to the network [19]. Each selective activation neuron, k, possesses an importance factor, m_k^p, that defines its relevance according to the pattern (sample), p, presented to the network. The output of these neurons, y_k^p, is given by (6):

$$y_k^p = m_k^p \mathcal{F}_k(a_k^p) = m_k^p \mathcal{F}_k \left(\sum_j w_{jk} y_j^p + \theta_k \right), \tag{6}$$

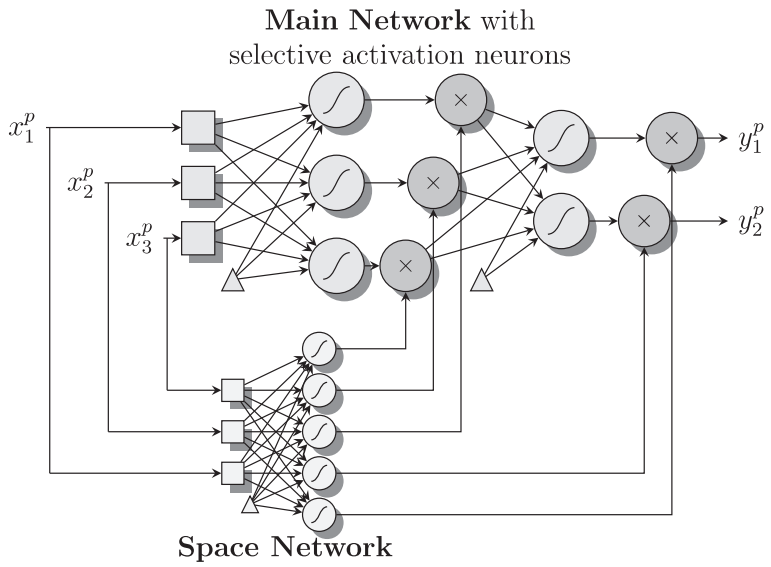

Fig. 2 *Example of a Multiple Feed-Forward (MFF) network. Squares represent inputs, darker circles (with the symbol ×) multipliers, lighter circles neurons, and triangles the bias.*

where $\mathcal{F}_k$ is the neuron activation function, a_k^p its activation, θ_k the bias and w_{jk} the weight of the connection between neuron j and neuron k. The farther from zero m_k^p is the more important the neuron contribution becomes. On the other hand, a value of zero means the neuron is completely irrelevant for the network output and one can interpret such a value as if the neuron is not present in the network.

In Lopes and Ribeiro, Multiple Feed-Forward (MFF) networks, which create a seamless partition of the input space, were proposed [19] yielding good results in several applications such as financial data analysis [20], analysis of market orientation [21] and character recognition [22]. MFF networks, represented in Figure 2, combine a main network containing selective activation neurons with a space network responsible for determining their importance factors. Both networks encompassing an MFF are trained together as a whole using the Multiple Back-Propagation (MBP) algorithm, which can be considered a generalization of the Back-Propagation (BP) algorithm. Usually this architecture presents better generalization properties than traditional feed-forward networks trained with the BP algorithm [23].

MBP uses the same rule for updating the weights as the BP algorithm, i.e. after presenting a given pattern p the network, the weights are adjusted by (7):

$$\Delta_p w_{jk} = \gamma \delta_k^p y_j^p + \alpha \Delta_l w_{jk} \ , \tag{7}$$

where γ is the learning rate, δ_k^p the local gradient of neuron k, $\Delta_l w_{jk}$ the weight change for the last pattern l and α the *momentum* term. However, the equations of the local gradient for the output, o, and hidden, h, neurons, given respectively

by (8) and (9), differ from the standard neuron equations:

$$\delta_o^p = (d_o^p - y_o^p) m_o^p \mathcal{F}_o^{'}(a_o^p) \ , \tag{8}$$

$$\delta_h^p = m_h^p \mathcal{F}_h^{'}(a_h^p) \sum_{o=1}^{N_o} \delta_o^p w_{ho} \ . \tag{9}$$

Notice however that if we consider all neurons to be equally important regardless of the pattern presented to the network, i.e. if we consider all the m_k^p constant and equal to 1, then equations (6) to (9) are identical to the corresponding BP equations.

Let us consider that $\boldsymbol{\chi} = \left\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^p, \ldots, \mathbf{x}^n\right\}$, where $\mathbf{x}^p$ denotes the input vector of pattern p, given by the p row of $\boldsymbol{\chi}$. Similarly let us consider that $\boldsymbol{\tau} = \left\{\mathbf{r}^1, \mathbf{r}^2, \ldots, \mathbf{r}^p, \ldots, \mathbf{r}^n\right\}$, where $\mathbf{r}^p$ denotes the response vector of pattern p, given by the p row of $\boldsymbol{\tau}$. We may view an NN as a non-linear mapping function, f_Λ, which transforms an input vector $\mathbf{x}^p \in \mathbb{R}^d$ into an output vector $\mathbf{y}^p \in \mathbb{R}^c$ containing the associated outputs ($f_\Lambda : \mathbf{x} \mapsto \mathbf{y}$), where c represents the number of outputs (classes). Typically f_Λ has a set of t parameters $(w_1, w_2, \ldots, w_t)$ (network weights) which are adjusted during the training procedure, seeking to minimize the errors between the actual output provided by the network and the desired outputs for the samples in the training dataset.

Assume that r_j^p is a random variable with Bernoulli distribution representing the act of obtaining the value of x_j^p ($r_j^p \sim Be(q_j)$). In order to deal with the missing data values we propose transforming the values x_j^p by taking into consideration r_j^p:

$$\tilde{x}_j^p = f(x_j^p, r_j^p) \ . \tag{10}$$

This transformation can be carried out by a neuron, k, with selective activation (named selective input), containing a single input, x_j^p, and an importance factor m_k^p identical to r_j^p, in which case (10) can be rewritten as (11) using (6):

$$\tilde{x}_j^p = r_j^p \mathcal{F}_k(w_{jk} x_j^p + \theta_k) \ . \tag{11}$$

If the value x_j^p cannot be obtained then the selective input associated to it will behave as if it does not exist, since r_j^p will be zero. On the other hand, if the value of x_j^p is available ($r_j^p = 1$), the selective input will actively participate in the computation of the network outputs. This can be viewed as if there are two different models, bound to each other, sharing information. One model for the case where the value of x_j^p is known and another one for the case where it cannot be obtained (is missing). Fig. 3 shows the physical model (NSIM) of a network containing a selective input and the two conceptual models inherent to it. A network with N selective inputs will have 2^N different models bonded to each other and constrained in order to share information (network weights). This can be viewed as if we decompose f_Λ into sub-functions that share information (parameters) among each other. For the simpler case of a network having a single selective input, f_Λ could be decomposed into two different functions, $f_{\Lambda 1}(w_1, w_2, w_3, \ldots, w_s)$ and $f_{\Lambda 2}(w_1, w_2, w_3, \ldots, w_s, w_{s+1}, \ldots, w_t)$, that share s parameters (for the network presented in Fig. 3, s corresponds to the number of weights of the Model 1. It is guaranteed that all the models share at least s parameters, s being equal to the

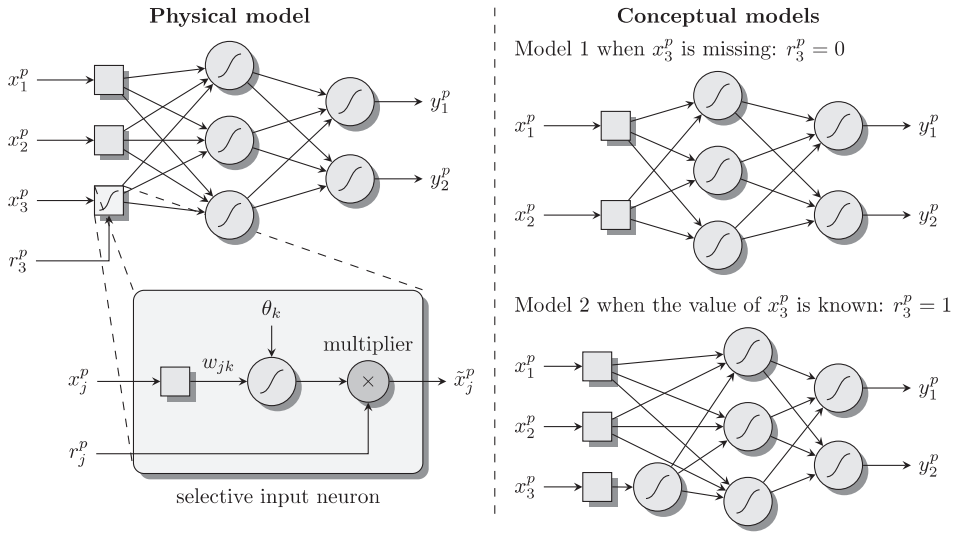

Fig. 3 *Physical and conceptual models of a network with a selective input ($k = 3$).*

number of weights that the network would have if the inputs with MVs were not considered at all [10].

Although conceptually there are multiple models, from the point of view of the training procedure there is a single model (NSIM), f_Λ with t adjustable parameters (weights). When a pattern is presented to the network, only the parameters directly or indirectly related to the inputs with known values are adjusted (observe equations (8) and (9)). Thus, only the relevant (conceptual) models will be adjusted [10].

The NSIM presents a high degree of robustness, since it is prepared to deal with faulty sensors. If the system which integrates the NSIM realizes a given sensor has stopped working it can easily deactivate (discard) all the models inherent to that specific sensor, by setting $r_j^p = 0$. Thus, consequently the best model available for the remaining sensors working properly will be considered. In addition, the NSIM does not require MAR or MCAR assumptions to hold, since only the known data is used actively to build the model.

We have implemented a standalone and a Graphics Processing Unit (GPU) version of the NSIM, integrated with the BP and MBP algorithms. The GPU version can be found both in the Multiple Back-Propagation software and in the GPU Machine Learning Library (GPUMLib) [24]. The MBP software which also implements the standalone version is available at `http://dit.ipg.pt/MBP` or alternatively at `http://sourceforge.net/projects/mbp/`, while GPUMLib can be obtained at `http://gpumlib.sourceforge.net/`.

Database	n	d	c	Proportion of MVs (%)	Features with MVs		MVs per feature (avg. %)	(stdev. %)
Annealing	898	47	5	49.22	37	(78.72%)	62.52	37.04
Audiology	226	93	24	2.85	23	(24.73%)	11.54	22.43
Breast cancer	699	9	2	0.25	1	(11.11%)	2.29	0.00
Congressional	435	16	2	5.63	16	(100.00%)	5.63	5.41
Hepatitis	155	19	2	5.67	15	(78.95%)	7.18	11.05
Horse colic	368	92	2	15.26	59	(64.13%)	23.79	16.01
Japanese credit	690	42	2	0.97	32	(76.19%)	1.27	0.24
Mammographic	961	5	2	3.37	5	(100.00%)	3.37	3.22
Mushroom	8124	110	2	1.11	4	(3.64%)	30.53	0.00
Soybean	683	77	19	8.73	76	(98.70%)	8.85	6.03

Tab. I *Main characteristics, proportion and distribution of the MVs for the UCI database benchmark experiments, after data pre-processing. Note that the average (avg.) and the standard deviation (stdev.) of MVs per feature does not include features without MVs.*

4. Results and Discussion

4.1 Experimental Setup

To evaluate the performance of the NSIM, we carried out extensive experiments on several UCI databases [25] with different distributions and proportion of MVs. Tab. I presents the main characteristics of the databases and an overview of the proportion and distribution of the MVs in each database, after pre-processing.

For our experiments we use 5-fold stratified cross validation partitions and for statistical significance 30 different networks were trained in each partition for each method and algorithm used.

The results of the NSIM were compared with single imputation and multiple imputation methods. Multiple imputation is considered one of the most powerful approaches for estimating MVs [18]. For single imputation the version 3.7.5 of the Weka software package was used [26]. For multiple imputation the NORM (Multiple imputation of incomplete multivariate data under a normal model) software was used [27]. NORM uses the Expectation-Maximization (EM) algorithm either with the maximum-likelihood or the posterior mode estimates. Since the maximum-likelihood estimate fails for many of the databases in Tab. I, the posterior mode was used. The EM algorithm is a technique for fitting models to incomplete data, by capitalizing on the relationship between the unknown parameters associated to the data model and the missing data itself [7].

4.2 Benchmarks Results

Tab. II presents the macro-average F-Measure performance (%) according to the algorithms used to train the NNs and the methods chosen for handling the MVs. As expected, the MBP algorithm performs better than BP regardless of the method used to handle MVs. On average the F-Measure performance of MBP excels the one of BP by 0.20%, 0.50% and 0.82% respectively for single imputation, multiple

	NSIM		Single Imputation		Multiple Imputation	
Database	**BP**	**MBP**	**BP**	**MBP**	**BP**	**MBP**
Annealing	97.13±02.69	**97.77±02.55**	91.93±06.79	93.22±06.65	93.04±07.40	93.16±06.53
Audiology	56.46±14.16	58.64±13.08	53.73±14.56	55.33±13.25	56.76±14.16	**61.07±14.72**
Breast cancer	95.05±01.59	95.42±01.69	95.38±01.21	**95.53±01.67**	95.01±01.53	95.37±01.66
Congressional	93.20±02.07	94.30±01.58	93.84±01.86	94.12±02.24	**94.83±01.79**	94.70±01.46
Hepatitis	70.10±06.23	73.55±05.99	72.63±07.96	72.20±07.53	**75.89±10.35**	75.44±09.98
Horse colic	84.31±02.56	84.86±02.44	83.29±02.80	83.11±02.78	87.30±02.06	**87.37±02.30**
Japanese credit	**81.98±02.45**	81.50±02.81	81.43±02.53	81.07±02.25	81.27±01.93	81.59±02.45
Mammographic	**81.62±01.74**	81.07±01.97	79.78±02.28	78.23±02.56	79.93±02.00	79.45±02.36
Mushroom	99.97±00.02	99.96±00.02	**99.98±00.02**	**99.98±00.01**	99.97±00.02	**99.98±00.01**
Soybean	93.43±00.68	**94.34±00.94**	91.63±01.26	92.87±00.85	92.54±00.68	93.48±00.66

Tab. II *Macro-average F-Measure performance (%) according to the methods used to handle the MVs and the algorithms used to train the NNs.*

imputation and NSIM methods. Using the Wilcoxon signed rank test, for the case of the NSIM, the null hypothesis of BP having an equal or better F-Measure than the MBP algorithm is rejected at a significance level of 0.05.

Comparing our method with the use of single imputation, we can verify that our method outperforms single imputation both for the BP and MBP algorithms, respectively by 0.96% and 1.58% on average. This considerable gain in terms of F-Measure performance, especially in the case of the MBP algorithm, is validated by Wilcoxon signed rank test: the null hypothesis of the MBP models created using single imputation having an equal or better F-Measure than those with the NSIM is rejected at a significance level of 0.01.

In contrast with single imputation, multiple imputation yields better results than the NSIM, concerning the BP algorithm (+0.33% on average). However, if we make use of the statistical evidence and adopt MFF networks trained with the MBP algorithm, then both approaches will perform similarly (on average multiple imputation performs better than NSIM by less than 0.02%).

Note however that if we consider only the datasets for which the proportion of MVs represents at least 5% of the whole data (annealing, congressional, hepatitis, horse colic and soybean), then concerning the MBP algorithm, our method excels the multiple imputation on average by 0.14%. These results seem intuitive since in principle multiple imputation works better when the proportion of MVs is smaller, in which case more data are available for validating the estimates inferred. These results assume particular relevance, if we consider that the appropriate choice of the method for handling MVs is especially important when the fraction of MVs is large [18].

Another important consideration is the impact that the proportion of features with MVs has on each method. For example if we look at the datasets containing a high-proportion of features with MVs then the F-Measure performance of the NSIM is once again superior to the corresponding performance of multiple imputation. Considering only the datasets for which at least 3/4 of the features contain MVs (annealing, congressional, hepatitis, Japanese credit, mammographic mass and soybean) we can verify that, for the MBP algorithm, on average NSIM

improves the F-Measure performance by 0.79% relatively to multiple imputation. This shows that our model tends to perform better than multiple imputation when the MVs spread throughout a large proportion of the features.

Additionally the NSIM presents the best solution in terms of system integration, in particular for hardware realization. Multiple imputation requires the system to include not only the multiple imputation algorithm itself, but also all the data needed for computing the adequate estimates. While tools such as the MBP software can generate code for any NN, to our knowledge there are no such tools or open source code (in general purpose languages) which would allow to easily embed multiple imputation in their NN systems. Moreover, the time and memory constraints necessary for the imputation process to take place would in many cases render such systems useless.

5. Conclusions and Future Work

In this paper we present a solution to the MVP that involves the integration of an NSIM into NNs allowing them to cope directly with MVs. To our best knowledge this is the first method that allows NNs, otherwise considered to be highly sensitive to MVs [28], to cope directly with this ubiquitous problem without requiring data to be pre-processed. Thus, NNs turn out to be positioned as an excellent alternative to (other) algorithms capable of dealing directly with the MVP (e.g. decision trees, rule extractors).

Through the use of selective inputs the proposed approach accounts for the creation of different conceptual models, while maintaining a single physical model. The NSIM works as if we divide the original training dataset into several subsets of data containing all the combinations of complete features, one for each set of maximal independent data without MVs, in order to create an ensemble of NNs. However, by using a single physical model, the training time of the NSIM is that of a single model (a fraction of the time that would be required to create the ensemble). Moreover, by independently training each subset of data we could attain models with significant discrepancies in terms of classification performance. Most likely those trained with a small fraction of the original samples would exhibit smaller performance than those trained with more samples. In this context the NSIM is also advantageous as it harmonizes the performance of all the models by forcing conceptual models to share as much parameters as possible.

The proposed solution presents several advantages as compared to traditional methods for handling MVs, making this a first-class method for dealing with this crucial problem: (*i*) it reduces the burden and the amount of time associated to the pre-processing task by not requiring the estimation of MVs; (*ii*) it preserves the uncertainty inherently associated to the MVP allowing the algorithms to differentiate between missing and real data; (*iii*) it does not require MAR or MCAR assumptions to hold, since only the known data are used actively to build the models; (*iv*) unlike pre-processing methods which may inject outliers into the data and cause undesirable bias, the NSIM uses the best conceptual model depending exclusively on the available data; (*v*) the NSIM can infer and take advantage of any informative information associated to the presence of MVs; (*vi*) it presents the best solution in terms of system integration, in particular for hardware realization as it

does not require the inclusion of additional and most likely complex systems; (*vii*) NSIM shows a high degree of robustness, since it is prepared to deal with faulty sensors. Deactivating the models inherent to that specific sensor and consequently using the best model available for the remaining working sensors is a simple operation; (*viii*) its classification performance, considering the MBP algorithm, is similar to state-of-the-art multiple imputation methods and the tests conducted show that the NSIM performs better than multiple imputation methods when the proportion of MVs is significant (more than 5% in our tests) or when the prevalence of MVs affects a large number of features.

Future work will exploit the possibility of using selective inputs on other types of NNs and extend this work to radial basis functions and recurrent networks.

References

[1] Kotsiantis S. B., Zaharakis I. D., Pintelas P. E.: Machine learning: a review of classification and combining techniques, Artificial Intelligence Review, Springer Netherlands, **26**, 3, 2006, pp. 159–190.

[2] Karhunen J.: Robust PCA Methods for Complete and Missing Data, Neural Network World, **21**, 2011, pp. 357–392.

[3] Tuikkala J., Elo L. L., Nevalainen O. S., Aittokallio T.: Missing value imputation improves clustering and interpretation of gene expression microarray data, BMC Bioinformatics, **9**, no. 202, 2008.

[4] Markeya M. K., Tourassi G. D., Margolis M., DeLong D. M.: Impact of missing data in evaluating artificial neural networks trained on complete data, Computers in Biology and Medicine, **36**, 5, 2006, pp. 516–525

[5] García-Laencina P. J., Sancho-Gómez J.-L., Figueiras-Vidal A. R.: Pattern classification with missing data: a review, Neural Computing & Applications, **19**, 2010, pp. 263–282.

[6] Bramer M. A.: Principles of data mining, Springer-Verlag, 2007.

[7] Nelwamondo F. V., Mohamed S., Marwala T.: Missing data: A comparison of neural network and expectation maximization techniques, Current Science, **93**, 11, 2007, pp. 1514–1521.

[8] Schafer J. L., Graham J. W.: Missing Data: Our View of the State of the Art, Psychological Methods, **7**, 2, 2002, pp. 147–177.

[9] Lopes N., Ribeiro B.: A robust learning model for dealing with missing values in many-core architectures, Proceedings of the 10th International Conference on Adaptive and Natural Computing Algorithms – Part II (ICANNGA 2011), LNCS 6594, Springer-Verlag, 2011, pp. 108–117.

[10] Lopes N., Ribeiro B.: A Strategy for Dealing With Missing Values by Using Selective Activation Neurons in a Multi-Topology Framework, The 2010 International Joint Conference on Neural Networks (IJCNN 2010), 2010, pp. 1–5.

[11] López-Molina T., Pérez-Méndez A., Rivas-Echeverría F.: Missing values imputation techniques for neural networks patterns, Proceedings of the 12th WSEAS International Conference on Systems (ICS 2008), World Scientific and Engineering Academy and Society, 2008, pp. 290–295.

[12] Mockus A.: Guide to Advanced Empirical Software Engineering, Chapter 7 – Missing Data in Software Engineering, Forrest Shull, Janice Singer and Dag I.K. Sjberg eds., Springer-Verlag London, 2008, pp. 185–200.

[13] Little R. J. A., Rubin D. B.: Statistical analysis with missing data. 2nd ed., Wiley, New Jersey, 2002.

[14] Serpen G.: A heuristic and its mathematical analogue within artificial neural network adaptation context, Neural Network World, **15**, 2005, pp. 129–136.

[15] Alavala C. R.: Fuzzy Logic and Neural Networks: Basic Concepts & Applications. New Age International Publishers, 2008.

[16] Wang S.: Classification with incomplete survey data: a Hopfield neural network approach. Computers & Operations Research, **32**, 2005, pp. 2583–2594.

[17] Aikl L. E., Zainuddin Z.: A Comparative Study of Missing Value Estimation Methods: Which Method Performs Better?, Proceedings of the International Conference on Electronic Design (ICED 2008), 2008, pp. 1–5.

[18] Ayuyev V. V., Jupin J., Harris P. W., Obradovic Z.: Dynamic Clustering-Based Estimation of Missing Values in Mixed Type Data, Proceedings of the 11th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2009), LNCS 5691, Springer-Verlag, 2009, pp. 366–377.

[19] Lopes N., Ribeiro B.: Hybrid learning in a multi-neural network architecture. Proceedings of the International Joint Conference on Neural Networks (IJCNN 2001), **4**, 2001, pp. 2788–2793.

[20] Bucur L.: Exploring Chaos with Sparse Kernel Machines, Proceedings of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2010), 2010, pp. 239–242.

[21] Silva M., Moutinho L., Coelho A., Marques A.: Market orientation and performance: modelling a neural network, European Journal of Marketing, **43**, 3/4, 2009, pp. 421–437.

[22] Chacko B. P., Babu Anto P.: Character Recognition using Multiple Back Propagation Algorithm, Proceedings of the National Conference on Image Processing, 2010, pp. 209–212.

[23] Lopes N., Ribeiro B.: An efficient gradient-based learning algorithm applied to neural networks with selective actuation neurons, Neural, Parallel & Scientific Computations, Dynamic Publishers, Inc., Atlanta, USA, **11**, 2003, pp. 253–272.

[24] Lopes N., Ribeiro B.: GPUMLib: An Efficient Open-Source GPU Machine Learning Library, International Journal of Computer Information Systems and Industrial Management Applications, **3**, 2011, pp. 355–362.

[25] Frank A., Asuncion A.: UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, `http://archive.ics.uci.edu/ml`

[26] Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I. H.: The WEKA Data Mining Software: An Update; SIGKDD Explorations, **11**, 1, 2009, pp. 10–18.

[27] Schafer J. L.: NORM: Multiple imputation of incomplete multivariate data under a normal model, version 2, `http://www.stat.psu.edu/~jls/misoftwa.html`, 1999.

[28] Ye N. (editor): The handbook of data mining, Lawrence Erlbaum Associates, 2003.