Expert Systems with Applications 40 (2013) 1333–1341

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Classifying patterns with missing values using Multi-Task Learning perceptrons

Pedro J. García-Laencina [a,*], José-Luis Sancho-Gómez [b], Aníbal R. Figueiras-Vidal [c]

[a] *Centro Universitario de la Defensa de San Javier (University Centre of Defence at the Spanish Air Force Academy), MDE-UPCT, Calle Coronel Lopez Peña, s/n, 30720 Santiago de la Ribera, Murcia, Spain*
[b] *Universidad Politécnica de Cartagena, Department of Information and Communications Technologies, Plaza del Hospital 1, 30202 Cartagena, Murcia, Spain*
[c] *Universidad Carlos III de Madrid, Department of Signal Theory and Communications, Avda. de la Universidad 30, 28911 Leganés, Madrid, Spain*

A R T I C L E I N F O

Keywords:
Pattern classification
Missing values
Imputation
Multi-Task Learning
Multi-Layer Perceptron

A B S T R A C T

Datasets with missing values are frequent in real-world classification problems. It seems obvious that imputation of missing values can be considered as a series of secondary tasks, while classification is the main purpose of any machine dealing with these datasets. Consequently, Multi-Task Learning (MTL) schemes offer an interesting alternative approach to solve missing data problems. In this paper, we propose an MTL-based method for training and operating a modified Multi-Layer Perceptron (MLP) architecture to work in incomplete data contexts. The proposed approach achieves a balance between both classification and imputation by exploiting the advantages of MTL. Extensive experimental comparisons with well-known imputation algorithms show that this approach provides excellent results. The method is never worse than the traditional algorithms – an important robustness property – and, also, it clearly outperforms them in several problems.

1. Introduction

Missing values are a frequent practical inconvenient in pattern classification problems (García-Laencina, Sancho-Gómez, & Figueiras-Vidal, 2010; Little & Rubin, 2002; Saar-Tsechansky & Provost, 2007; Schafer, 1997). Its causes are diverse, from human errors to monitoring or communication failures (Lakshminarayan, Harp, & Samad, 2004, 2000), including lack of measurement equipments or that their use is not appropriate (Jerez et al., 2010; Markey & Patel, 2004).

The way of dealing with the missing values may have relevant effects on incomplete datasets classification performance. A first basic approach is to train classifiers only with complete cases. Obviously, this means to discard useful information and does not solve the classification of new incomplete instances. There are also embedded procedures which can directly deal with unknow input values without imputation, such as decision trees (Quinlan, 1993) and fuzzy neural networks (Lim, Leong, & Kuan, 2005). However, the more extended procedures impute the missing values. Imputation is the most widely-used approach in many real applications with missing data because the great majority of decision making tools can not be directly used for incomplete data classification. Besides, imputation also provides additional information (imputed values) which can be used to enhance the classification performance. This work is focused on missing data estimation solutions based on machine learning procedures, that are the most promising approaches to solve practical inference problems.

The most popular imputation procedures, whose principles were first applied two decades ago, use a separate process to estimate the missing values. This process can be based on a statistical model for the class distributions, such as the well-known Gaussian Mixture Models (GMMs) with an Expectation–Maximization (EM) formulation for Maximum-Likelihood (ML) estimation of both the mixture parameters and the missing values (Delalleau, Courville, & Bengio, 2008; Ghahramani & Jordan, 1993; Zio, Guarnera, & Luzi, 2007), or any other reasonable scheme taking into account the proximity or similarity of the incomplete instances to other complete examples, such as the K-Nearest Neighbors (KNN) (Batista & Monard, 2003; García-Laencina, Sancho-Gómez, Figueiras-Vidal, & Verleysen, 2009; Troyanskaya et al., 2001) and the Self-Organizing Maps (SOMs) (Fessant & Midenet, 2002; Latif & Mercier, 2010, chap. 12) imputation methods. Auxiliary learning machines, such as Multi-Layer Perceptrons (MLPs) (Gupta & Lam, 1996; Nishanth, Ravi, Ankaiah, & Bose, 2012; Silva-Ramírez, Pino-Mejías, López-Coello, & Cubiles-de-la Vega, 2011) or Support Vector Machines (SVMs) (Mallinson & Gammerman, 2003), can also be used to estimate missing values. Nevertheless, although all these approaches effectively use the available information, they do not consider directly that the final objective is just to obtain a good classification performance.

An extreme possibility is to impute values by minimizing the surrogate misclassification error during the classifier training (Yoon & Lee, 1999). This is not appropriate because overfitting

* Corresponding author. Tel.: +34 968189979; fax: +34 968188780.
E-mail address: pedroj.garcia@cud.upct.es (P.J. García-Laencina).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.08.057

effects on missing data imputation may appear, since the unknown values are estimated considering not more than the classification objective and forgetting the statistical structure of the samples. In this case, imputed values may be unlikely estimates according to the input data density.

This paper will explore a new perspective for imputation: To provide appropriate estimates for the missing values with the purpose of improving the operative classification performance. On the one hand, it seems clear that the key task is to classify complete or incomplete new samples. On the other hand, in real scenarios, there is not possibility of measuring the quality of the imputation process because it is not possible to compare with the original lost values. However, the imputed values cannot distort the input data distribution and, consequently, unlikely estimates are not appropriate. Then, it would be interesting to find imputation methods taking into account that classification is the main task and, at the same time, providing accurate estimates for the missing values.

Multi-Task Learning (MTL) is a well established framework that naturally fits these purposes. It takes advantage of the similarity of some tasks to solve them simultaneously (Caruana, 1997; Evgeniou, Micchelli, & Pontil, 2005; Ghosn & Bengio, 2003; Silver, 2000). As said above, this is just the point of view that we will adopt to solve classification problems with missing data. In particular, we will analyze here the use of a modified traditional architecture for MTL, a Multi-Layer Perceptron (MLP) with common hidden units for all the tasks by proposing new procedures to train this network and to classify new samples on missing data situations. As it has been analyzed in our previous studies and experiments (García-Laencina, Sancho-Gómez, & Figueiras-Vidal, 2006; García-Laencina, Serrano-García, Figueiras-Vidal, & Sancho-Gómez, 2007; Sancho-Gómez, García-Laencina, & Figueiras-Vidal, 2009), MTL offers the possibility of training MLP schemes that have an output for solving the decission problem – main classification task – and a set of additional outputs that give imputed values for the missing features – secondary imputation tasks.

It is trivial to establish an adequate (problem dependent) compromise between the main and the secondary tasks. As this work will show, this compromise can be achieved by weighting the corresponding costs and apply simple Cross-Validation (CV) techniques to find appropriate weight values for the problem under analysis. Consequently, MTL allows to consider both the requirements of getting accurate classification performance and simultaneously providing imputed values by paying attention to the original input data structure and avoiding overfitting. Although MTL algorithms have evolved from their original MLP-type forms (Caruana, 1997; Silver, 2000) to more sophisticated versions that are used in kernel machines (Evgeniou et al., 2005) or Gaussian processes (Bonilla, Chai, & Williams, 2008), the aim of this work is to shown that the MTL principles are useful for missing data imputation and, then, we will focus on MLP-type architectures for the sake of clarity. So, we will present relevant extensions of our preliminary work on this subject (García-Laencina et al., 2006; García-Laencina et al., 2007; García-Laencina, Figueiras-Vidal, & Sancho-Gómez, 2008; Sancho-Gómez et al., 2009). In particular,

- to implement the previously discussed idea of weighting the main and the secondary tasks by means of a convex combination of their surrogate costs, which allows a very easy CV for choosing the convex combination parameter;
- to use surrogate costs that are selected according to the specific demands of incomplete data classification problems;
- to introduce a new mechanism to classify new samples that limits overfitting effects on missing data imputation.

All these aspects, including the proposed MTL scheme, will be extensively analyzed in the next sections.

The remainder of this paper is structured as follows: Section 2 presents the details of the proposed MTL-based architecture for classification and imputation under missing values conditions. Its training is explained in Section 3, while the operative classification stage is described in Section 4. Results of applying the proposed method to solve some synthetic and real problems are shown in Section 5, comparing them with the performances of a number of well established representative imputation methods. Finally, the main conclusions of our work and some suggestions for further research close the paper.

2. Proposed MTL architecture

In a supervised D-dimensional classification problem, the dataset has the form $\{\mathbf{x}_n, \mathbf{m}_n, \mathbf{t}_n\}$, $n = 1, \ldots, N$, where $\mathbf{x}_n$ is the nth pattern, a $D + 1$ column vector with components $x_{0n} = 1$ and x_{dn} a certain value or ? (unknown) if that component is empty; $\mathbf{m}_n$ is a $D + 1$ indicator vector with values m_{dn}, $1 \leqslant d \leqslant D$, equal to 1 or 0 if x_{dn} is present or not, respectively; and $\mathbf{t}_n$ is the label for the nth observation, which consists in an *1*-of-C binary vector for $C > 2$ classes, $\mathbf{t}_n = [t_1, \ldots, t_C]^T$, or just a bit if the problem has two classes. Without loss of generality, we will assume that the incomplete variables are $\{x_{d'}\}$ with $d' = 1, \ldots, D'$ and $D' \leqslant D$.

Following an MTL approach, this problem is composed of $D' + 1$ learning tasks: The main task and the D' secondary tasks. The main task is learning $\mathbf{t}$ from $\mathbf{x}$ and each secondary task consists of learning $x_{d'}$ from the remaining input features of $\mathbf{x}$ (excluding $x_{d'}$) and $\mathbf{t}$. To solve them within an MTL framework, this work will use, for the sake of clarity, a multi-task MLP architecture with one hidden layer of J units with hyperbolic tangent activations and an output layer of C softmax units (or a sigmoidal unit) for classification, plus D' linear units (as many as the number of incomplete input variables) for imputation (Sancho-Gómez et al., 2009). Hidden neurons are different to the classical neuron, because they compute different outputs for the distinct tasks to be learned. Thus, for a given input pattern, each hidden neuron provides an output value for the output nodes of the main task, and a different output value for the output node of every secondary task (imputation tasks). These are the outputs computed by the jth hidden unit:

- *Main hidden output*, z_{M_j}. An output which is connected to the classification output units, having the form:

$$z_{M_j} = \tanh\left(\mathbf{v}_j^T(\tilde{\mathbf{x}} \odot \mathbf{r}_M)\right) \tag{1}$$

  where $\tilde{\mathbf{x}} = [\mathbf{x}^T, \mathbf{t}^T]^T$ is the network input vector and $\mathbf{v}_j$ is a vector of common weights connecting the input units ($\tilde{\mathbf{x}}$) to the jth hidden unit. Besides, $\mathbf{r}_M = [1, \mathbf{1}^T, \mathbf{0}^T]^T$ is a vector which indicates that vector $\mathbf{t}$ is not taken into account for computing z_{M_j}. Thus, the main task is only learned using $\mathbf{x}$. Note that $\odot$ is the Kronecker (element-wise) product, and that $\tilde{\mathbf{x}}$, $\mathbf{v}_j$ and $\mathbf{r}_M$ have the same dimension, equal to $(D + 1) + C$.
- *Secondary hidden outputs*, $\{z_{S_{d'j}}\}_{d'=1}^{D}$. There is a different hidden output for each imputation unit:

$$z_{S_{d'j}} = \tanh\left(\mathbf{v}_j^T\left(\tilde{\mathbf{x}} \odot \mathbf{r}_{S_{d'}}\right)\right). \tag{2}$$

  $\tilde{\mathbf{x}}$ and $\mathbf{v}_j$ are as above, and the binary vector $\mathbf{r}_{S_{d'}}$ indicates the input units from $\tilde{\mathbf{x}}$ which are used to learn the d' secondary task. Since this task is associated to the learning of the incomplete variable $x_{d'}$, it is learned from all network inputs of $\tilde{\mathbf{x}}$ except $x_{d'}$ and, then, all components of $\mathbf{r}_{S_{d'}}$ are *1* except the corresponding to $x_{d'}$ which is zero.

Note that these hidden neurons do not include in the linear combination the input signal that they have to learn in the corre-

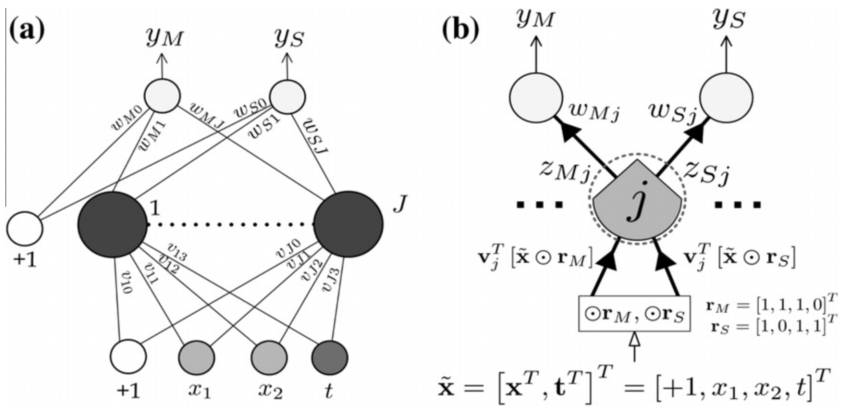
(a)
y_M
y_S
w_{M0}
w_{M1}
w_{MJ}
w_{S0}
w_{S1}
w_{SJ}
1
J
+1
v_{10}
v_{11}
v_{12}
v_{13}
v_{J0}
v_{J1}
v_{J2}
v_{J3}
+1
x_1
x_2
t
(b)
w_{Mj}
w_{Sj}
z_{Mj}
z_{Sj}
j
$\mathbf{v}_j^T[\tilde{\mathbf{x}} \odot \mathbf{r}_M]$
$\mathbf{v}_j^T[\tilde{\mathbf{x}} \odot \mathbf{r}_S]$
$\odot\mathbf{r}_M, \odot\mathbf{r}_S$
$\mathbf{r}_M = [1,1,1,0]^T$
$\mathbf{r}_S = [1,0,1,1]^T$
$\tilde{\mathbf{x}} = [\mathbf{x}^T, \mathbf{t}^T]^T = [+1, x_1, x_2, t]^T$

Fig. 1. MTL network for simultaneous imputation and classification in a two-class bidimensional problem, x_1 being incomplete. (a): MTL neural architecture; (b): The developed MTL hidden neuron, where $\mathbf{r}_M$ shows that y_M learns t from x_1 and x_2, and $\mathbf{r}_S$ shows that y_S learns x_1 from x_2 and t.

sponding output unit. This is done to avoid direct connections to map the input as output (Caruana, 1997; García-Laencina et al., 2007). It should be noted that each hidden node is not treated as two different standard hidden neurons (a different neuron for each task – classification and imputation), i.e., the network is not composed of two different hidden layers for imputation and classification. Thus, the proposed multi-task MLP scheme has a unique first-layer of common weights, the classification and imputation tasks being learned in parallel while using a shared representation (Caruana, 1997; Sancho-Gómez et al., 2009; Silver, 2000).

From (1) and (2), the network outputs are computed according to

$$y_{M_c} = \frac{\exp\left(\mathbf{w}_{M_c}^T \mathbf{z}_M\right)}{\sum_{c'=1}^{C} \exp\left(\mathbf{w}_{M_{c'}}^T \mathbf{z}_M\right)}, \quad c = 1, \ldots, C; \tag{3}$$

$$y_{S_{d'}} = \mathbf{w}_{S_{d'}}^T \mathbf{z}_{S_{d'}}, \quad d' = 1, \ldots, D'; \tag{4}$$

where $\mathbf{z}_M = \left[1, z_{M_1}, \ldots, z_{M_J}\right]^T$ is the extended hidden output vector for classification and $\mathbf{z}_{S_{d'}} = \left[1, z_{S_{d'1}}, \ldots, z_{S_{d'J}}\right]^T$ for imputation; $\mathbf{w}_{M_c} = [w_{M_{c0}}, \ldots, w_{M_{cJ}}]$ with $c = 1, \ldots, C$, and $\mathbf{w}_{S_{d'}} = \left[w_{S_{d'0}}, \ldots, w_{S_{d'J}}\right]$ with $d' = 1, \ldots, D'$ are the output weight vectors corresponding to the main and secondary output nodes, respectively.

Finally, in order to adequately explain the proposed MTL network, Fig. 1 shows the MTL-based MLP architecture to solve the simplest scenario: A binary classification task of bidimensional patterns and an incomplete input feature, x_1, for example. Main and secondary tasks are modeled by y_M and y_S, respectively. All hidden units are full-connected by the first-layer weights, and, as it has been mentioned above, the jth hidden neuron of the MTL network computes different outputs depending on the output unit to which the hidden output is directed (see Fig. 1b). In particular, z_{M_j} and z_{S_j} denote, respectively, the jth neuron output which simultaneously acts as an input for y_M and y_S. In this case, $\mathbf{r}_M = [1,1,1,0]^T$ because y_M learns t from x_1 and x_2, and $\mathbf{r}_S = [1,0,1,1]^T$ because y_S learns x_1 from x_2 and t. Thus, from (1) and (2), the hidden outputs are $z_{M_j} = \tanh(v_{j0} + v_{j1}x_1 + v_{j2}x_2)$ and $z_{S_j} = \tanh(v_{j0} + v_{j2}x_2 + v_{j3}t)$, respectively.

3. Training stage

After initializing the weights to small random values and the missing values to initial imputations coming from any easy method, such as the hot-deck procedure (each missing value of an input vector is imputed from its closest training complete pattern) (Little & Rubin, 2002; Schafer, 1997), the proposed architecture has to be trained to minimize a reasonable cost function E. As a first approach (Sancho-Gómez et al., 2009), a simple additive function to define the global error could be used: $E = E_M + E_S$, where E_M is the main task cost and E_S is an average of the secondary tasks costs. In this case, both tasks have the same importance during training, i.e., learning is balanced and without any task priority. Nevertheless, it is desirable to use a selective transfer method, which regulates the impact of each task error, because classification and imputation do not have the same relevance. For these reasons, this work introduces the following cost function:

$$E = \lambda E_M + (1 - \lambda) E_S, \tag{5}$$

where λ is a convex combination parameter which determines the relative importance of the secondary tasks with respect to the main one and how much information is transferred between tasks. Consequently, the permissible value for this parameter is $0 < \lambda < 1$. The role of λ is just to modulate the preference for the classification task with respect to imputations. Since E_M and E_S are not homogeneous costs, there is not any rule of thumb for the selection of λ value: The best value of λ can be at any point between 0 and 1, and it depends on the nature of the problem being addressed. An appropriate design value for λ can be selected by Cross Validation (CV), as it will be explained later.

Depending on the type of task to be learnt, different choices of the cost function can be considered. For classification tasks, the goal is to model the posterior probabilities of class membership conditioned on the input variables. For this reason, the cross entropy error (Cid-Sueiro, Arribas, Urbán-Munoz, & Figueiras-Vidal, 1999), is used for E_M:

$$E_M = -\sum_{n=1}^{N} \sum_{c=1}^{C} t_{cn} \ln y_{M_{cn}}, \tag{6}$$

It is important to remark that, as E_M is jointly minimized with E_S, y_{M_c} is not properly estimating an a posterior class probability. Nevertheless, it is an appropriate and reasonable indicator of the class to be selected.

On the other hand, for imputation tasks, it seems reasonable to give the same importance to each one of the secondary tasks:

$$E_S = \frac{1}{D'} \sum_{d'=1}^{D'} E_{S_{d'}} \tag{7}$$

where $E_{S_{d'}}$, with $d' = 1, \ldots, D'$, are the costs for each imputation task. As it is widely known and some previous works have shown (García-Laencina et al., 2008; Sancho-Gómez et al., 2009), the use of the squared error may not provide satisfactory results, because it has sensitivity to high absolute values and it provides mean

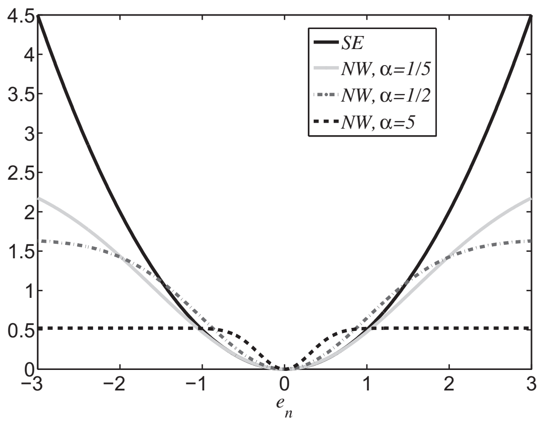

Fig. 2. Squared Error (SE) and Normalized Welsch (NW) cost functions for different values of α: $\frac{1}{5}$, $\frac{1}{2}$, 5. For low values of α, the NW cost tends to the SE, whereas, for high values of α, it tends to the MAP cost.

estimates for the incomplete feature values which may not preserve the input data distribution. Thus, we will use the "normalized" Welsch cost (Huber, 2009) for secondary tasks:

$$E_{S_{d'}} = \frac{1}{N}\sum_{n=1}^{N}\beta\left[1 - \exp\left(-\alpha e_n^2\right)\right], \tag{8}$$

where $e_n = y_{Sd'n} - x_{d'n}$ is the conventional estimation error and $\beta = \sqrt{2\alpha}\exp\left(-\frac{1}{2}\right)$ is selected to ensure that the absolute value of the derivative of (8) does not exceed the unity. As it is shown in Fig. 2, cost (8) tends to the squared error for low values of α, and, when α is high, to the Maximum A Posteriori (MAP) or "hit or miss" cost. In order to avoid the above mentioned sensitivity problems and obtain plausible imputed values, it is enough to start with a low value for α and, then, training the MLP until convergence. In the next stage, α is increased and the MTL network continues being trained from the previous stage. This process is repeated until α reaches a high enough value to produce a MAP-based estimation for missing data. This iterative minimization of E_S, using (7) as the error function, provides robust and accurate imputed values that preserve the input data distribution. Needless to say, the components of (8) corresponding to missing values are excluded from the training computations because it is not possible to compute e_n when $\mathbf{x}_n$ is incomplete on its dth attribute.

With respect to the minimization of (5), we use the scaled conjugate gradient method (Moller, 1993) that is computationally efficient and widely used to train MLP architectures. While output weights are updated considering its specific task error derivatives, common input weights are influenced by the contributions from all tasks. Besides, learning is early stopped using cross-validation (Prechelt, 1998).

Another important issue is that each secondary output is used to estimate and fill in the missing values in the corresponding incomplete input feature during learning. In particular, imputation is done when any learning task is paralyzed, i.e., when the change in any training error function (E_M or E_S), measured in a sequence of training epochs, falls below a certain threshold (Bishop, 1995; Prechelt, 1998). After this, the MTL network is trained again with the whole edited dataset (i.e., the complete data portion and the incomplete data portion with imputed attributes) until a saturation training error level occurs or the cross-validation error indicates overtraining. Since classification and imputation tasks are learned by the same MTL network, the classification task learning affects to these imputed values, and so, this imputation is oriented to solve the classification task. Imputed values are those that contribute to improve the classification accuracy.

4. Operation stage

If a new sample has not missing values, the classification part of the trained machine directly gives the class to which that sample belongs. When there are missing values, the network operation is different. Firstly, classification output values cannot be appropriately calculated because there are unknown input values. Secondly, the imputation cannot be done because the sample label is not available.

In order to solve this "circular" difficulty, we propose to apply a "principle of consistency". We mean that the preferred solution must show the best fitting to the classifier design principles. Let us see how this principle can be implemented.

If the 1-of-C label code corresponding to each class is used as an input to the MTL network, we can obtain the corresponding imputation estimates for the missing variables. Then, the classification output of the network is computed with these imputed values. If the sample comes from a given class, it seems reasonable that the imputed pattern (the observed input values plus the estimated missing values) assuming that class will tend to indicate the same class in a clearer form than the inputs including the estimated values assuming other classes.

Thus, given an incomplete input pattern $\mathbf{x}$, the process will consist on:

1. Obtaining a complete input vector, $\hat{\mathbf{x}}^c$, for each c-class by filling the missing values in $\mathbf{x}$ with the corresponding secondary outputs given by the network for an input $\tilde{\mathbf{x}}$ composed of $\mathbf{x}$ and the c-class label, and then,
2. Classifying the pattern by means of

$$\hat{c} = \arg\max_c\{y_{M_c}(\hat{\mathbf{x}}^c)\}. \tag{9}$$

Note that although this criterion selects the most consistent class, it does not consider the quality of the imputed values. Given that the proposed method has to properly solve both tasks (imputation and classification), a quality measure of the imputation has to be included in the consistency criterion. Considering the class conditional probability density function (pdf) value of $\mathbf{x}$ as a quality imputation measure, the proposed consistency rule is given by

$$\hat{c} = \arg\max_c\{y_{M_c}(\hat{\mathbf{x}}^c)\hat{p}(\hat{\mathbf{x}}^c|c)\}, \tag{10}$$

where $\{\hat{p}(\hat{\mathbf{x}}^c|c)\}_{c=1}^{C}$ are any kind of reasonable estimates of sample probability densities for class c. Here, these estimates are computed using Parzen window method (Archambeau, Valle, Assenza, & Verleysen, 2006; Härdle, Müller, Sperlich, & Werwatz, 2004; Parzen, 1962) from the corresponding imputed training samples. In general, Gaussian windows give satisfactory results, using the Leave-One-Out (LOO) validation method to select the dispersion parameter (Härdle et al., 2004).

In comparison with (9), (10) incorporates $\hat{p}(\hat{\mathbf{x}}^c|c)$, which acts as weighting factor for the corresponding classification outputs. Thus, if an imputed pattern for a given class is unlikely according to its pdf value, the probability of being assigned to that class is decreased, and vice versa.

5. Experiments

We have carried out a series of experiments with two kinds of problems: Datasets without missing values (introducing them artificially), to check the effects of different missing situations; and datasets with missing values, to assess how the proposed method works in real incomplete problems. In particular, the tested complete problems are a synthetic *Toy* dataset and two well-known datasets, *Iris* (Frank & Asuncion, 2010) and *Telugu* (Pal & Majumder,

Table 1
Summary of datasets' characteristics.

Dataset	Type	*D*	*C*	*N*
Toy	S	2	2	1500
Iris	RC	4	3	150
Telugu	RC	3	6	871
Sick-Thyroid	RI	5	2	1500
Smoking	RI	8	3	3549
Pima Indians	RI	7	2	662

1977). Note that, in these three problems, the missing values are created by purely random mechanisms, with different Missing Rates (MR) in order to measure their influence on the classification accuracy. With respect to the remaining three incomplete datasets, *Sick-Thyroid* and *Pima-Indians* are from the UCI repository (Frank & Asuncion, 2010), and *Smoking* becomes from (Bull, 1994, chap. 13). Table 1 shows the main parameters of these six problems. For each dataset, it gives the type (*S*, synthetic, *RC*, real complete, *RI*, real incomplete), the number of input attributes (*D*), the number of classes (*C*) and the total number of samples (*N*). More details of each dataset will be given later on its corresponding section. As it can be seen from Table 1, the analyzed datasets are diverse and representative to evaluate the usefulness of the proposed approach. It is important to note that, for complete datasets, this work analyzes different missing data scenarios, i.e., different missing percentages on different input features, in order to evaluate the missing data influence on the obtained classification accuracy, which is one of the key aspects on incomplete data classification problems.

In all the cases, the performances of the MTL network have been compared with four well-known imputation procedures that, in general, give good results:

- K Nearest Neighbors imputation (*KNNimp*), as described in (Batista & Monard, 2003; García-Laencina et al., 2009; Troyanskaya et al., 2001). First, the nearest, most similar, neighbors are found by minimizing a distance function. Then, the estimates for missing values are computed as the weighted average of the *K* nearest samples with no unknown data in the same variable to be imputed.
- Self Organizing Map imputation (*SOMimp*), as described in (Fessant & Midenet, 2002, 2010). First, when an incomplete pattern is presented to the SOM, its nearest node is chosen ignoring the distances in the missing variables; secondly, an activation group composed of neighbor nodes is selected; and finally, each imputed value is computed based on the weights of the activation group of nodes in the missing dimensions.
- Multi-Layer Perceptron imputation (*MLPimp*), as described in (Gupta & Lam, 1996; Silva-Ramírez et al., 2011). *MLPimp* consists on training an MLP using only the complete cases as a regression model: Given *D* input features, each incomplete attribute is learned (it is used as output) by means of the $(D-1)$ other attributes given as inputs. The MLP outputs are used to impute unknown values given the observed ones. When several attributes are missing, several MLP schemes have to be designed, one per missing variables combination.
- Gaussian Mixture Model imputation (*GMMimp*), as described in Ghahramani and Jordan (1993), Zio et al., 2007, and Delalleau et al., 2008. The input data distribution is estimated by a GMM trained with the Expectation–Maximization (EM) algorithm. Missing features can be treated naturally in this framework, by computing the expectation of the sufficient statistics for the expectation, or E-step, over the missing values as well as the mixture components. Thus, at the E-step, the missing values are estimated given the observed data and the current estimate of the model parameters. In the maximization, or M-step, the likelihood function is maximized under the assumption that the missing values are known. The estimates of the missing values from the E-step are iteratively used for imputation until convergence.

In order to carry out fair comparisons with the proposed method, the resulting imputed datasets from the above four methods are used to learn the classification task. It is done by training one hidden layer MLP classifiers whose number of hidden neurons are selected by means of stratified 10-fold cross-validation (CV) process applied with 30 different training runs (i.e., different network weight initializations). The cost function for these MLP classifiers is the cross-entropy error and its maximum number of hidden neurons is 20. It is important to note that the weight initialization for each MLP classifier is the same for all tested procedures and, also, for our proposed network. With respect to the optimization, as it has been already mentioned, we use the Scaled Conjugate Gradient (SCG) method (Moller, 1993). The SCG algorithm uses a numerical approximation for the second derivatives (Hessian matrix), but it avoids instability by combining the model-trust region approach from the Levenberg–Marquardt algorithm with the conjugate gradient approach. For a detailed explanation, see (Moller, 1993). The SCG method only requires to initialize two parameters: σ, which determines the weight change for the second derivative approximation; and η, that regulates the indefiniteness of the Hessian. According to Moller (1993) and Nabney (2002), we initially set them to $\sigma = 1e-4$ and $\eta = 1$.

The same 10-fold CV procedure is used to select the non-trainable parameters of the traditional imputation techniques. The explored values are:

- *KNNimp*. For the number of nearest neighbors: 1–20;
- *SOMimp*. For the number of nodes: 2–200;
- *MLPimp*. For the number of hidden neurons: 1–20;
- *GMMimp*. For the number of Gaussians: 2–20.

The parameters in these four approaches are selected by considering the obtained performance for the imputation task in the validation sets. The number of hidden neurons for the MLP classifier is also chosen by considering the classification performance in the validation sets. According to the literature (García-Laencina et al., 2010; Saar-Tsechansky & Provost, 2007), the imputation accuracy is usually measured by computing the root mean square error between the imputed values and the 'true' values, and it can only be done when missing values are artificially inserted. Whereas, in real incomplete datasets, it is not possible to measure the quality of imputed data because its original 'true' values are unknown. Due to this, in these situations, we select those combination of parameters for each imputation method and the MLP classifier that gives better classification results in the validation set.

With respect to the proposed method, the MTL-based MLP architectures are also trained by means of the SCG method. Both the convex combination parameter (λ) and the number of hidden neurons (J) are selected by means of the same 10-fold CV as above, with 30 different initializations for weights. Explored values are:

- for λ: 0.1–0.9, in 0.1 steps;
- for J: 1–20.

Parzen windows exploration with LOO (Archambeau et al., 2006) covers the dispersion parameter from 0.01 to 2.0, in 0.01 steps. Note that, in all experiments, the consecutive values of normalized Welsch cost parameter (α) are $\frac{1}{5}$, $\frac{1}{3}$, $\frac{1}{2}$, 1, 2, 3 and 5.

Table 2
Test classification accuracy (in%, mean ± standard deviation) for the dataset *Toy*. A: The classical imputation methods (hidden units of the MLP classifier in brackets). B: Variants of the proposed network (hidden layer size and values of λ in brackets).

	MR (%)	KNN_{imp}	SOM_{imp}	MLP_{imp}	GMM_{imp}
A	5	90.7 ± 0.55 (9)	90.9 ± 0.44 (10)	90.2 ± 0.40 (10)	**91.7 ± 0.25** (8)
	10	89.6 ± 0.55 (11)	90.6 ± 0.62 (10)	89.2 ± 0.55 (11)	**91.0 ± 0.35** (9)
	20	88.1 ± 0.67 (10)	89.6 ± 0.36 (7)	88.7 ± 0.47 (11)	**89.7 ± 0.27** (8)
	30	86.7 ± 0.36 (11)	86.4 ± 0.50 (8)	87.4 ± 0.43 (9)	**87.5 ± 0.61** (12)
	40	85.3 ± 0.36 (11)	85.0 ± 0.66 (5)	86.5 ± 0.34 (11)	**86.7 ± 0.66** (11)
	MR (%)	SE-AP	SE-JP	NW-AP	NW-JP
B	5	91.1 ± 0.47 (6,0.7)	91.7 ± 0.35 (5,0.7)	91.5 ± 0.46 (5,0.7)	**92.3 ± 0.23** (6,0.8)
	10	90.8 ± 0.61 (6,0.7)	90.9 ± 0.49 (5,0.7)	91.1 ± 0.35 (5,0.7)	**91.2 ± 0.46** (6,0.8)
	20	89.3 ± 0.47 (5,0.8)	89.7 ± 0.30 (5,0.8)	89.8 ± 0.57 (6,0.7)	**90.0 ± 0.44** (5,0.6)
	30	86.7 ± 0.56 (6,0.7)	87.5 ± 0.39 (6,0.7)	87.6 ± 0.40 (5,0.8)	**87.9 ± 0.52** (6,0.6)
	40	85.9 ± 0.62 (5,0.7)	86.1 ± 0.53 (4,0.6)	86.3 ± 0.55 (6,0.8)	**86.6 ± 0.62** (5,0.6)

5.1. Experimental results

5.1.1. Toy

This dataset is a modification of the well-known Ripley's problem (Ripley, 1996). *Toy* is a two-dimensional synthetic classification problem composed of five Gaussian components. The mean vectors for each one of the five components are $\mu_1 = (1,1)^T$, $\mu_2 = (-0.7,0.3)^T$, $\mu_3 = (0.3,0.3)^T$, $\mu_4 = (-0.3,0.7)^T$, $\mu_5 = (0.4,0.7)^T$; the covariances are all isotropic: $\mathbf{\Sigma}_j = 0.03\mathbf{I}$. All five components have the same number of observations. The first three Gaussians belongs to the first class and the remaining two components to the another class. We generate three groups of 500 samples for training, validation (no CV is applied) and testing. x_1 suffers the artificial missing process.

We will present here the 30 run average results corresponding to using the Squared Error (SE) or the Normalized Welsch (NW) error for learning the imputation task. Besides, during the operation mode, the consistency mechanism is applied considering the two presented approaches: Consistency based only in "A Posteriori" (AP) estimated class probabilities (9) and the consistency based on the "Joint Probability" (JP) (10). We will indicate these methods by *SE-AP*, *SE-JP*, *NW-AP* and *NW-JP*. Table 2 shows the experimental results for the classical imputation procedures (A) and for the variants of the proposed MTL method (B). Best results for each block are in boldface. As a reference, the CV optimal MLP classifier (6 hidden neurons) with entropic cost offers a 30-run average classification accuracy of 92.0% with a standard deviation of 0.36 for complete data.

It must be emphasized that the effects of even high missing rates are moderate because the problem has a simple structure. It clearly appears that *GMMimp* has a systematic advantage with respect to the other standard imputation methods, although many differences are statistically irrelevant for high MRs. With respect to the proposed method, the *NW-JP* approach also presents a minor but systematic advantage. Finally, the *NW-JP* form of the proposed method near systematically outperforms the best of the classical imputation techniques, *GMMimp*, although the statistical relevance of the differences dissappears with increasing values of MR.

5.1.2. Iris

This well-known dataset consists of 50 samples from each of three classes, and four input features are measured from each instance (Frank & Asuncion, 2010). From the 150 samples, 50 are randomly chosen for the testing set and the remaining 100 samples are used during the training stage. A CV sized MLP (with 5 hidden units) gives a 30-run average classification accuracy of 96.8% and standard deviation 1.65 for the complete dataset.

In this problem, we will also check the differences that appear when missing values correspond to different variables. Obviously, the effects of missing data will be more important when they affect variables that are more related to the classification task. This can be measured by the Normalized Mutual Information (NMI) between each feature and the target class considering them as random variables X_d and T, respectively. Thus,

$$NMI(X_d;T) = \frac{I(X_d;T)}{\sum_{d'=1}^{D} I(X_{d'};T)}, \tag{11}$$

where $I(X_d;\ T)$ is the mutual information measured between the dth input attribute and the target class:

$$I(X_d;T) = H(T) - H(T|X_d), \tag{12}$$

H being the entropy:

$$H(T) = \sum_{c=1}^{C} P(T=c)\log\frac{1}{P(T=c)}. \tag{13}$$

$$H(T|X_d) = \int_{X_d} p(x_d)\sum_{c=1}^{C} P(T=c|x_d)\log\frac{1}{P(T=c|x_d)}\mathrm{d}x_d. \tag{14}$$

These entropies are estimated by using relative frequencies for $P(T = c)$ and Parzen windows of circularly simmetric forms for $p(X_d|T = c)$, that are used in (14) by applying Bayes formula and replacing the integral in this equation by a summation over the training samples of the factor of $p(x_d)$ (Kwak & Choi, 2002). In this work, the NMI estimation based on the Parzen window approach is computed using the software developed in Peng, Long, and Ding (2005).

For a classification problem, the NMI measures the amount of information contained in an input feature for predicting the target class variable. Thus, the higher the value of $NMI(X_d;\ T)$, the more relevant the dth input feature for the classification task. In particular, for the *Iris* problem, $NMI(X_1;\ T) = 0.20$, $NMI(X_2;\ T) = 0.07$, $NMI(X_3;\ T) = 0.39$ and $NMI(X_4;\ T) = 0.34$. According to these values, it is interesting to analyze how missing rates affect to a relatively unimportant variable, such as x_1, and to an important component, such as x_4, for example.

Table 3 shows the classification accuracy (in %, mean ± standard deviations) for the four traditional imputation procedures and the proposed method (in its *NW-JP* form; i.e., using the normalized Welsch cost for learning the imputation task and the consistency based on the joint densities for classification of new incomplete samples). Overall best values are in boldface and best values for the classical methods are in italic. When missing values appear in x_1, which is relatively unimportant for classification purposes, differences among all the procedures are irrelevant. Low degradations for high MR values and the high values of λ that are found by CV for the proposed *MTLimp* method support that x_1 values are of low importance. If missing values affect to x_4, both the faster degradation and the lower values of λ indicate that this feature is more important – although not so much for causing significant dif-

Table 3
Test classification accuracy percentages (± standard deviations) for the four standard imputation techniques and the proposed MTL method in the problem *Iris*.

Attribute	MD (%)	KNN$_{imp}$	SOM$_{imp}$	MLP$_{imp}$	EM$_{imp}$	MTL$_{imp}$
x_1	5	96.7 ± 1.62 (4)	96.9 ± 1.60 (3)	96.7 ± 1.69 (5)	96.6 ± 1.71 (3)	**96.9 ± 1.26 (4,0.9)**
	10	96.7 ± 1.77 (5)	96.7 ± 1.78 (4)	96.4 ± 1.81 (6)	96.4 ± 1.65 (4)	**96.8 ± 1.31 (6,0.9)**
	20	96.6 ± 1.56 (4)	96.7 ± 1.53 (5)	96.3 ± 1.65 (5)	96.7 ± 1.67 (4)	**96.7 ± 1.36 (6,0.8)**
	30	96.5 ± 1.66 (6)	96.6 ± 1.50 (4)	96.1 ± 1.70 (6)	96.3 ± 1.59 (4)	**96.7 ± 1.22 (6,0.9)**
	40	96.4 ± 1.58 (6)	96.5 ± 1.61 (5)	96.0 ± 1.69 (6)	96.1 ± 1.60 (4)	**96.6 ± 1.31 (6,0.9)**
x_4	5	96.7 ± 1.57 (4)	96.9 ± 1.28 (3)	96.7 ± 1.21 (4)	96.6 ± 1.37 (4)	**97.1 ± 1.30 (5,0.6)**
	10	96.2 ± 1.61 (4)	96.5 ± 1.46 (5)	96.3 ± 1.31 (5)	96.2 ± 1.30 (4)	**97.0 ± 1.27 (4,0.6)**
	20	96.0 ± 1.67 (5)	96.1 ± 1.38 (5)	96.0 ± 1.35 (4)	96.2 ± 1.32 (3)	**96.5 ± 1.21 (6,0.7)**
	30	95.7 ± 1.50 (5)	95.8 ± 1.67 (6)	95.9 ± 1.37 (4)	95.8 ± 1.40 (4)	**96.2 ± 1.29 (5,0.7)**
	40	95.1 ± 1.76 (5)	95.1 ± 1.74 (6)	95.5 ± 1.59 (6)	95.2 ± 1.45 (5)	**95.9 ± 1.38 (6,0.6)**

Table 4
Test classification accuracy percentages (± standard deviations) for the four standard imputation techniques and the proposed MTL method in the problem *Telugu*.

MR (%)	KNN$_{imp}$	SOM$_{imp}$	MLP$_{imp}$	GMM$_{imp}$	MTL$_{imp}$
5	84.1 ± 1.26 (12)	83.7 ± 1.13 (14)	83.8 ± 1.19 (16)	83.6 ± 1.19 (12)	**84.7 ± 0.59 (16,0.8)**
10	83.1 ± 1.17 (14)	83.1 ± 1.18 (16)	83.3 ± 1.04 (14)	82.9 ± 1.00 (14)	**83.8 ± 0.90 (14,0.7)**
20	81.2 ± 1.29 (14)	80.7 ± 1.23 (16)	81.0 ± 1.20 (18)	80.5 ± 1.12 (14)	**81.9 ± 1.00 (14,0.8)**
30	79.6 ± 1.36 (16)	78.1 ± 1.36 (18)	79.2 ± 1.19 (18)	78.7 ± 1.23 (16)	**80.8 ± 1.05 (16,0.7)**
40	77.8 ± 1.11 (14)	76.1 ± 1.21 (18)	77.5 ± 1.21 (16)	76.9 ± 1.35 (14)	**78.7 ± 0.99 (18,0.7)**

ferences among the imputation procedures we are comparing. In any case, it must be outlined that *MTLimp* is always as good as the best alternative procedure, a robustness property which has practical interest.

5.1.3. Telugu

This data set consists of 871 patterns (Pal & Majumder, 1977). There are six overlapping vowel classes (Indian Telugu vowel sounds) and three input features (formant frequencies). 30% of samples are chosen for testing and the remaining patterns are used during the design stage. This first random division has been done with the restriction that both sets contain approximately the same proportions of class labels as the original dataset. An MLP classifier of 14 hidden neurons (selected by CV) obtains a classification accuracy equal to 84.82 ± 1.09% (average of 30 runs). The normalized MI between each input feature and the target class is evaluated: $NMI(X_1;\ T) = 0.32$, $NMI(X_2;\ T) = 0.42$, and $NMI(X_3;\ T) = 0.26$. In this problem, input values are randomly removed in the most relevant feature (x_2). Table 4 shows the classification results.

It can be seen that loosing x_2 values produces a serious performance degradation. Besides, when the MR increases, the proposed method *MTLimp* has a severe advantage with respect to the traditional imputation procedures. Among these traditional procedures, *KNNimp* gives the best results for this dataset.

5.1.4. Sick thyroid

This is a binary (sick/healthy) problem dataset (Frank & Asuncion, 2010) whose observations have 21 binary attributes and seven continuous variables, from which 6 are laboratory test results. There are 2800 training and 972 test samples. From the observed variables, the five laboratory results with missing values are chosen for our experiments and, then, we have to deal with a hard scenario: A dataset plenty of missing values where all attributes are incomplete. Table 5 shows the MR for training and test sets.

Applying again the traditional techniques and the proposed method in the same way as above, the results of Table 6 (row A) are obtained. It can be seen that *MTLimp* provides a slight advantage.

5.1.5. Smoking

In this dataset (Bull, 1994), there are three classes, eight features, 1927 training and 928 test instances. Variables x_4 and x_5 are missing for 14.1% and 15.3% of the samples, respectively. Results of the corresponding experiments are shown in row B of Table 6. Once again, there is a minor advantage of the proposed method with respect to the traditional techniques.

Table 5
MRs percentages in the Sick-Thyroid dataset.

	x_1 (%)	x_2 (%)	x_3 (%)	x_4 (%)	x_5 (%)
Training	10.2	20.9	6.6	10.6	10.5
Test	8.7	18.9	4.8	9.3	9.3

5.1.6. Pima Indians

This dataset is available at (Ripley, 1996). It contains seven medical measurements on 632 patients and a binary label which indicates whether diabetes appeared or not for each individual. The dataset consists of 300 training patterns, with 100 incomplete cases, and a test set of 332 complete instances. Variables x_3, x_4 and x_5 are incomplete, with MRs of 4.3%, 32.7% and 1%, respectively. Table 6 (in row C) shows the corresponding experimental results. Note that, in this problem, the advantage of *MTLimp* is very clear.

5.2. Discussion of the results

For the first three complete problems, we can say that the proposed MTL-based MLP network provides a performance advantage when the rate of missing values or the importance of the affected features are high enough to entail relatively high difficulties to solve the classification task. When missing data effects are less important, all the methods tend to give similar results, but it must be remarked that the proposed approach is at least as good as any other method in every situation. This kind of robustness and the possibility of getting some advantages under hard conditions make the proposed method interesting for its practical application.

With respect to the three incomplete problems, it has been checked that, in practical cases, the characteristic of robustness (it is never worse than the classical procedures) and the potential of getting better results of the proposed method appear again.

Table 6
Test classification results (in%, mean ± standard deviation) for (A) Sick-Thyroid, (B) Smoking and (C) Pima-Indians.

	KNNimp	SOMimp	MLPimp	GMMimp	MTLimp
A	96.9 ± 0.29 (15)	96.6 ± 0.33 (19)	96.8 ± 0.31 (18)	96.4 ± 0.31 (15)	**97.3 ± 0.33 (17,0.8)**
B	69.4 ± 0.25 (13)	69.3 ± 0.40 (13)	69.3 ± 0.43 (15)	69.2 ± 0.41 (12)	**69.6 ± 0.37 (15,0.6)**
C	76.9 ± 0.17 (14)	77.0 ± 2.06 (13)	76.5 ± 1.94 (15)	76.6 ± 1.97 (14)	**80.0 ± 1.03 (15,0.7)**

Table 7
Test classification accuracy (in%, mean ± standard deviation) of MTLimp vs. Parameter λ values for the three real incomplete problems. Hidden layer size is shown in brackets.

λ	Sick Thyroid	Smoking	Pima Indians
0.1	96.6 ± 0.40 (18)	68.8 ± 0.44 (12)	76.9 ± 1.65 (15)
0.2	96.7 ± 0.38 (16)	68.8 ± 0.41 (13)	77.0 ± 1.70 (15)
0.3	96.8 ± 0.45 (15)	69.0 ± 0.39 (12)	77.4 ± 1.67 (16)
0.4	96.7 ± 0.36 (16)	69.3 ± 0.46 (14)	77.8 ± 1.39 (14)
0.5	96.9 ± 0.31 (18)	69.4 ± 0.31 (13)	78.6 ± 1.58 (15)
0.6	97.0 ± 0.33 (16)	*69.6 ± 0.37 (15)*	79.9 ± 1.44 (16)
0.7	97.0 ± 0.36 (17)	**69.7 ± 0.36 (14)**	***80.0 ± 1.03 (15)***
0.8	***97.3 ± 0.28 (17)***	69.6 ± 0.30 (15)	79.2 ± 1.21 (15)
0.9	97.1 ± 0.31 (17)	69.3 ± 0.39 (15)	78.5 ± 1.32 (15)

Therefore, we can repeat our statement: Using the proposed MTL-based method is a really interesting practical alternative.

Finally, we will discuss now the sensitivity to CV selected values for the proposed method in the three real incomplete problems. We will focus on the convex combination parameter λ, since it is specific of the MTL-based network and it is important because its selection seems to be the reason to obtain the robustness characteristic that we have remarked above: This offers a problem oriented balance between imputation and classification. Table 7 shows the performance results of the proposed method for the above real problems when different values of λ are preselected. Best results are in boldface, and those corresponding to CV selection of values for λ are in italics. It can be observed that CV of λ provides more than reasonable results. The best value for *Sick Thyroid* and *Pima Indians*, and the next lower value, without any relevant degradation, for *Smoking*. Thus, CV seems to be an adequate method to select λ (and other non-trainable parameters) in the design of the proposed MTL networks.

6. Conclusions and further work

A Multi-Task Learning (MTL) based approach using Multi-Layer Perceptron (MLP) networks to impute missing values in classification problems has been presented. Unlike other traditional procedures, classification and imputation are combined in only one neural architecture, where classification is used as the main task and the imputation of each incomplete feature as a secondary task. All tasks are learned in parallel by the proposed MTL network. By doing so, classification guides the imputation process during the parallel learning of all tasks, while the aim of the secondary tasks is to provide imputed values which help to obtain accurate results for the main classification task. This method controls the relative importance of classification and imputation tasks during the MTL process by means of a convex combination of the corresponding cost terms, whose combination parameter is easily determined by Cross-Validation (CV). The proposed method is a form of getting a reasonable classification-oriented imputation mechanism. Classification of incomplete samples is carried out by finding the hypothesis that best fits the underlying model.

Extensive experimental results show that the proposed method has a valuable robustness property, since it is never worse than other well-known and widely used imputation techniques. The proposed approach also shows the capacity of providing better results than these traditional procedures when the effects of missing values are considerable. These properties make the proposed method very attractive for practical applications.

A clear research direction to improve these MTL-based networks is weighting each imputation task according to its classification relevance and/or missing values rate. Other ongoing work involves to add 'private' subnetworks for learning the specific regularites of each task. The effectiveness of other architectures that have been applied for MTL, such as kernel-type machines (Evgeniou et al., 2005), to solve incomplete data problems needs also to explored. Obviously, the extension to regression problems is of potential practical interest. Finally, we are analyzing the application of this framework to deal with semisupervised learning scenarios.

References

Archambeau, C., Valle, M., Assenza, A., & Verleysen, M. (2006). Assessment of probability density estimation methods: Parzen window and finite Gaussian mixtures. In *IEEE intl. symp. on circuits and systems (ISCAS)* (pp. 3245–3248). Island of Kos, Greece.

Batista, G. E., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence, 17*, 519–533.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press.

Bonilla, E., Chai, K. M., & Williams, C. (2008). Multi-task gaussian process prediction. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.). *Advances in neural information processing systems* (20, pp. 153–160). Cambridge, MA: MIT Press.

Bull, S. (1994). Analysis of attitudes toward workplace smoking restrictions. In N. Lange, L. Ryan, L. Billard, D. Brillinger, L. Conquest, & J. Greenhouse (Eds.), *Case studies in biometry* (pp. 249–271). New York, NY: Wiley.

Caruana, R. (1997). *Multitask learning*. Ph.D. Thesis, School of Computer Science, Carnegie Mellon University.

Cid-Sueiro, J., Arribas, J. I., Urbán-Munoz, S., & Figueiras-Vidal, A. R. (1999). Cost functions to estimate a posteriori probabilities in multiclass problems. *IEEE Transactions on Neural Networks, 10*, 645–656.

Delalleau, O., Courville, & A., Bengio, Y. (2008). Gaussian mixtures with missing data: An efficient EM training algorithm. In *Proc. of computing res. association conf.* (p. 155). Snowbird, UT.

Evgeniou, T., Micchelli, C., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research, 6*, 615–637.

Fessant, F., & Midenet, S. (2002). Self-organising map for data imputation and correction in surveys. *Neural Computing and Applications, 10*, 300–310.

Frank, A., & Asuncion, A. (2010). *UCI machine learning repository*. <http://archive.ics.uci.edu/ml>.

García-Laencina, P. J., Sancho-Gómez, J. -L., & Figueiras-Vidal, A. R. (2006). Pattern classification with missing values using multitask learning. In *Proc. intl. joint conf. on neural networks (IJCNN)* (pp. 3594–3601). Vancouver, Canada.

García-Laencina, P. J., Serrano-García, J., Figueiras-Vidal, A. R., & Sancho-Gómez, J. -L. (2007). Multi-task neural networks for dealing with missing inputs. In *Proc. intl. work-conf. on the interplay between natural and artificial computation (IWINAC)* (Vol. 4527, pp. 282–291). Murcia, Spain: LNCS. La Manga del Mar Menor.

García-Laencina, P. J., Figueiras-Vidal, A. R., & Sancho-Gómez, J. -L. (2008). Incomplete pattern classification using a multi-task approach. In *e-Proceedings of the 12th world multi-conference on systemics, cybernetics and informatics* (pp. 1–6). Orlando, Florida, USA.

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

García-Laencina, P. J., Sancho-Gómez, J.-L., Figueiras-Vidal, A. R., & Verleysen, M. (2009). K-Nearest neighbours with mutual information for simultaneous classification and missing data imputation. *Neurocomputing, 72*(7–9), 1483–1493.

Ghahramani, Z., & Jordan, M. I. (1993). Supervised learning from incomplete data via an EM approach. In J.D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information proc. sys* (vol. 6, pp. 120–127). Denver, CO.

Ghosn, J., & Bengio, Y. (2003). Bias learning, knowledge sharing. *IEEE Transactions on Neural Networks, 14*, 748–765.

Gupta, A., & Lam, M. S. (1996). Estimating missing values using neural networks. *Journal of the Operational Research Society, 47*, 229–238.

Härdle, W., Müller, M., Sperlich, S., & Werwatz, A. (2004). *Nonparametric and semiparametric models*. Berlin: Springer.

Huber, P. J. (2009). *Robust statistics* (2nd ed.). Hoboken, NJ: Wiley.

Jerez, J. M., Molina, I., García-Laencina, P. J., Alba, E., Ribelles, N., Martín, M., et al. (2010). Missing data imputation using statistical and machine learning approaches in a real breast cancer problem. *Artificial Intelligence in Medicine, 50*, 105–115.

Ji, C., & Elwalid, A. (2000). Measurement-based network monitoring: Missing data formulation and scalability analysis. In *Proc. IEEE intl. symp. on information theory* ((pp. 78)). Sorrento, Italy.

Kwak, N., & Choi, C.-H. (2002). Input feature selection by mutual information based on Parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*, 1667–1671.

Lakshminarayan, K., Harp, S., & Samad, T. (2004). Imputation of missing data in industrial databases. *Applied Intelligence, 11*, 259–275.

Latif, B. A., & Mercier, G. (2010). Self-organizing maps. In *Tech open, self-organizing maps for processing of data with missing values and outliers: Application to remote sensing images* (pp. 189–210).

Lim, C.-P., Leong, J.-H., & Kuan, M.-M. (2005). A hybrid neural network system for pattern classification tasks with missing features. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*, 648–653.

Little, R. J. A., & Rubin, D. B. (2002). *Statistical analysis with missing data* (2nd Edition). Hoboken, NJ: Wiley.

Mallinson, H., & Gammerman, A. (2003). *Support vector machine imputation methodology*. Tech. rep., Euredit Project WP 5.6, University of London.

Markey, M., & Patel, A. (2004). Impact of missing data in training artificial neural networks for computer-aided diagnosis. In *Proc. intl. conf. on machine learning and applications* (pp. 351–354). Louisville, KY.

Moller, M. F. (1993). *Efficient training of feed-forward neural networks*. Ph.D. Thesis, Computer Science Dept., Aarhus University.

Nabney, I. T. (2002). *NETLAB: Algorithms for pattern recognition*. London, UK: Springer.

Nishanth, K. J., Ravi, V., Ankaiah, N., & Bose, I. (2012). Soft computing based imputation and hybrid data and text mining: The case of predicting the severity of phishing alerts. *Expert Systems with Applications, 39*(12), 10583–10589.

Pal, S. K., & Majumder, D. D. (1977). Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Transactions on Systems, Man and Cybernetics, 7*, 625–629.

Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics, 33*, 1065–1076.

Peng, H., Long, F., & Ding, C. H. Q. (2005). Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*, 1226–1238.

Prechelt, L. (1998). Automatic early stopping using cross validation: Quantifying the criteria. *Neural Networks, 11*(4), 761–767.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA.

Ripley, B. D. (1996). *Pattern recognition and neural networks*. New York, NY: Cambridge University Press. <http://www.stats.ox.ac.uk/pub/PRNN/>.

Saar-Tsechansky, M., & Provost, F. (2007). Handling missing values when applying classification models. *Journal of Machine Learning Research, 8*, 1623.

Sancho-Gómez, J.-L., García-Laencina, P. J., & Figueiras-Vidal, A. R. (2009). Combining missing data imputation and pattern classification in a multi-layer perceptron. *Intelligent Automation and Soft Computing, Special issue on Soft Computing & Signal Processing, 15*, 539–553.

Schafer, J. L. (1997). *Analysis of incomplete multivariate data*. Boca Raton, FL: Chapman & Hall.

Silva-Ramírez, E.-L., Pino-Mejías, R., López-Coello, M., & Cubiles-de-la Vega, M.-D. (2011). Missing value imputation on missing completely at random data using multilayer perceptrons. *Neural Networks, 24*(1), 121–129.

Silver, D. L. (2000). *Selective transfer of neural network task knowledge*. Ph.D. Thesis, Canada: Department of Computer Science, University of Western Ontario.

Troyanskaya, O., Cantor, M., Alter, O., Sherlock, G., Brown, P., Botstein, D., et al. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics, 17*, 520–525.

Yoon, S.-Y., & Lee, S.-Y. (1999). Training algorithm with incomplete data for feed-forward neural networks. *Neural Processing Letters, 10*, 171–179.

Zio, M. D., Guarnera, U., & Luzi, O. (2007). Imputation through finite gaussian mixture models. *Computational Statistics & Data Analysis, 51*(11), 5305–5316.