2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Pattern Classification with Missing Values using Multitask Learning

Pedro J. García-Laencina, *Student Member, IEEE* , José-Luis Sancho-Gómez, and Aníbal R. Figueiras-Vidal, *Senior Member, IEEE*

***Abstract*— In many real-life applications it is important to know how to deal with missing data (incomplete feature vectors). The ability of handling missing data has become a fundamental requirement for pattern classification because inappropriate treatment of missing data may cause large errors or false results on classification. A novel effective neural network is proposed to handle missing values in incomplete patterns with Multitask Learning (MTL). In our approach, a MTL neural network learns in parallel the classification task and the different tasks associated to incomplete features. During the MTL process, missing values are estimated or imputed. Missing data imputation is guided and oriented by the classification task, i.e., imputed values are those that contribute to improve the learning. We prove the robustness of this MTL neural network for handling missing values in classification problems from UCI database.**

I. Introduction

Pattern classification methods based on Artificial Neural Networks (ANNs) have been successfully applied in many domains requiring intelligence, from medical diagnosis to fault detection in industrial machinery and speech recognition. ANNs can recognize patterns working simultaneosly with continuous, binary, ordinal and nominal data. A common problem in pattern recognition is the presence of *missing data*. Traditional classification methods usually cannot deal with real-world data, because of most of them ignore the presence of missing values in input patterns, *i.e.*, it is assumed that input patterns are complete. The problem of missing data arises in several fields of real-life applications. The different reasons for missing data can be ranging from sensor failures in engineering applications to non response in a survey [1], [2]. A clear example of the importance of handling missing data is that 45% of UCI data sets have missing values. Therefore, the ability of handling missing or uncertain inputs is essential in real application tasks of pattern classification because of inappropriate treatment of missing data may cause large errors or false results on classification.

This paper proposes a novel approach for handling and estimating missing values in classification problems using *Multitask Learning* (MTL). MTL was developed in 1993 by Rich Caruana [3]. The basic idea is that a task will be learned better if can leverage the information contained in the training signals of other related tasks during learning [4]. The task which is desired to be learnt better is called the primary or main task and the tasks whose training signals are used as hints by the main task are referred to as the secondary or extra tasks. Our method utilizes the incomplete features as extra tasks that are learned in parallel with the main classification task. Weights connections are dynamically adapted in function of the missing attributes for every input vector, independently of how missing data are distributed, and moreover, we use the outputs that learns incomplete features to estimate missing values during learning process. This missing data imputation is oriented by the learning of the classification task, in other words, it is oriented to solve the classification problem and these imputed values are those that contribute to improve the classification.

The remainder of this article is structured as follows: Section 2 presents the notation used in this work. In Section 3, an overview of missing data problem and basic approaches for handling missing values are described. In Section 4, it is shown both how MTL works and different neural architectures based on MTL. Proposed method is presented in Section 5 to solve a general classification problem with missing values. Next, in Section 6, our method is tested on real and artificial classification problems. Conclusions and future related works conclude the paper.

II. Pattern Classification with Incomplete Data

In general, classification problems normally involve the labeling of unclassified data with a specific output class, in other words, classification problems are seen as a learning a functional mapping from the input to output space [5].

$$f : \mathbf{X} \mapsto C \tag{1}$$

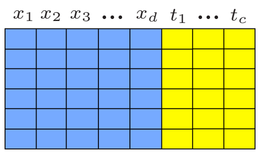
Each input pattern of $\mathbf{X}$ is associated to a specific class output belonging to one of c possible classes. In conventional classification tasks, all attribute values of each pattern are completely known and represented by a real vector $\mathbf{x}$, *i.e.*, input set $\mathbf{X}$ are completely observable. Let us consider that each input pattern $\mathbf{x}^{(n)}$ has d attribute real values, $\mathbf{x}^{(n)} = (x_1^{(n)}, x_2^{(n)}, ..., x_d^{(n)})$, and an output classification target $t^{(n)}$. Alternately, it is possible to code $t^{(n)}$ in a target vector $\mathbf{t}^{(n)}$ using a $1 - of - c$ codification, e.g., if there are five possible classes and the n-th pattern belongs

Pedro J. García-Laencina and José-Luis Sancho-Gómez are with the Departamento de Tecnologías de la Información y las Comunicaciones, Universidad Politécnica de Cartagena, 30202, Cartagena-Murcia, Spain (email: pedroj.garcia@upct.es, josel.sancho@upct.es).
Aníbal R. Figueiras-Vidal is with the Departamento de Teoría de Señal y Comunicaciones, Universidad Carlos III de Madrid, 28911, Leganés-Madrid, Spain (email: arfv@tsc.uc3m.es).

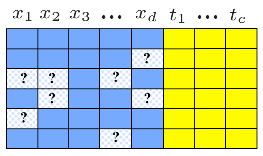
Authorized licensed use limited to: IEEE Xplore. Downloaded on February 10, 2009 at 08:25 from IEEE Xplore. Restrictions apply.

to the third one, its target vector will be $\mathbf{t}^{(n)} = (0, 0, 1, 0, 0)$.

In classification tasks discussed in this paper, input patterns can have some unknown attribute values (*i.e.*, missing values). Figure 1(a) shows a classical classification problem with complete dataset. On the other hand, Figure 1(b) shows a classification problem where some input vectors are incomplete. In this paper, we consider that missing values are not always in the same attribute among given samples (*e.g.*, it is possible that the *i*-th attribute value of one sample is missing while the same attribute of another example is known). We will refer to a missing value with ? symbol; thus, pattern $\mathbf{x}^{(4)} = (0.1, ?, -0.2, ?, 0.3)$ presents missing values at second and fourth attributes. In addition, we can define the *missing-data indicator matrix* $\mathbf{M} = (m_{ij})$, such that $m_{ij} = 1$ if $x_j^{(i)}$ is missing and $m_{ij} = 0$ if $x_j^{(i)}$ is present [1]. In the previous example, $\mathbf{m}^{(4)} = (0, 1, 0, 1, 0)$.



(a) A typical classification problem with a complete dataset.



(b) A classification problem with incomplete patterns, denoted by ? symbol.

Fig. 1. Two hypothetical classification problem of c classes. First, in (a), all patterns are completely known. In the other way, in (b), some input vectors present incomplete data.

III. Handling missing data

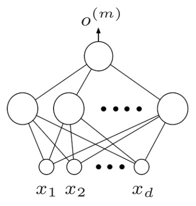
We will focus on methods for handling missing data by means of ANN in classification problems. In the literature, all proposed imputation procedures do not focus the estimation of missing values oriented to solve the classification task; their first aim is obtaining a complete data set and then an ANN learns the classification task using this complete data [6]–[8]. On the other hand, there are some methods that change the learning and the operation of an ANN to be able deal with missing inputs in classification problems [9]–[11].

In [6], Nordbotten uses models based on ANN for imputing survey variable values. In this work, a different ANN is trained to learn each incomplete feature being these networks used to realize the imputation task. Yoon *et al.* [7] suggests an algorithm that was composed of 3 steps, first of all, an ANN is trained with only the complete portion of dataset to learn the classification task; secondly, estimation of missing attributes in the incomplete cases with the trained network by error backpropagation is realized; finally, the ANN is re-trained with the whole dataset to learn the classification task. Markey *et al.* [8] analyzes the effect of missing data on trained ANN in three cases: without incomplete data, replacing the missing values using mean imputation and multiple imputation procedure.

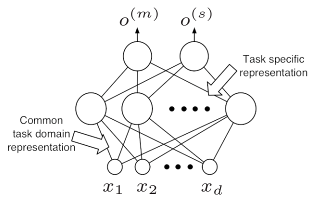
Other methods change its learning process to be able to deal with missing inputs. Ishibuchi *et al.* uses an interval representation of incomplete data with missing inputs [9], [10]. As input space is a d-dimensional inside an unit cube, each missing input was represented by an interval that includes its possible values, *i.e.*, $[0, 1]$. Learning of the proposed neural network is adapted to consider the interval representation in missing inputs. In [11], Viharos *et al.* develops a method for handling missing data based on the use of a validation flag for each input pattern. Validation flag indicates whether a value in the input vector is missing or not, and the inputs weights changes according this validation flag.

IV. Multitask Learning

Most approaches to machine learning focus on the learning of a single isolated task, *Single Task Learning* (STL). STL is used to refer to an ANN learning system that learns a single task. In order to explain a STL approach, consider a dataset $\mathcal{M}$, associated to a single (main) task, with its respective input set $\mathbf{X}^{(m)}$ and target set $\mathbf{T}^{(m)}$. Figure 2(a) shows STL scheme for solving this problem. This net can be trained to minimizing an error function between network outputs $o^{(m)}$ and target values $t^{(m)}$. Therefore, the network learns only a single task, in other words, the network learns only targets $\mathbf{T}^{(m)}$ from $\mathbf{X}^{(m)}$. Although, STL has been achieved great success, it overlooks basic details and advantages of human learning. Human learning frequently involves learning several tasks simultaneously; in particular, humans compare and contrast similar tasks for solving a problem. For example, if you want to learn periodic table, it is easier learning groups of related elements than learning the complete table.



(a) STL scheme.



$o^{(m)}$ $o^{(s)}$
Task specific representation
Common task domain representation
x_1 x_2 x_d

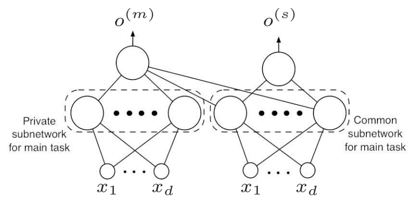
(b) MTL scheme.

Fig. 2. Standard net schemes. First, in (a), it is showed a STL network that learns only a main task from an input vector with d attributes. In (b), a MTL network learns a main task and a secondary task. This extra task helps to get a better performance in the learning of the main task.

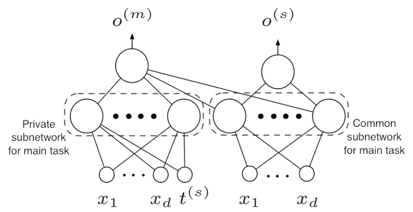
In last years, many works have applied these advantages to machine learning [12]–[17]. These works add extra related tasks to a main task and learn them at same time. This approach to learning is called *Multitask Learning* (MTL). MTL is a method that is designed to improve hypothesis generalization by transfering knowledge between tasks that are learned simultaneously, in parallel, in the same shared representational structure [3], [4]. Figure 2(b) shows this structure. The task which is desired to be learnt better is called the *main task* and the task whose training signals are used as hints by the main task are referred to as the *secondary tasks*. In order to explain a MTL approach, consider a dataset

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 10, 2009 at 08:25 from IEEE Xplore. Restrictions apply.

$\mathcal{M}$, associated to a main task, with its respective input set $\mathbf{X}^{(m)}$ and target set $\mathbf{T}^{(m)}$ and a dataset $\mathcal{S}$, associated to a secondary task, with its respective input set $\mathbf{X}^{(s)}$ and target set $\mathbf{T}^{(s)}$. In most of the cases, input data sets of all tasks are the same, $\mathbf{X}^{(m)} = \mathbf{X}^{(s)} = \mathbf{X}$. A first approach is to use only one network with a hidden layer of neurons for learning all tasks [15]. This sharing promotes inductive transfer: the hidden layer representations learned for the extra outputs are available to the main task output and often improve performance on the main task. With respect to weights, weights of the first layer are updated depending on the error of all tasks; while the weights, that connect each output unit to hidden neurons, are only influenced by errors due to the output corresponding task. The obvious disadvantages with MTL networks are the increased requirement for hidden nodes within the ANN and the longer training times that are required. Another disadvantage is that MTL systems, by default, assume that all tasks are related. This default assumption allows unrelated tasks to decrease the generalization performance accross all tasks causing a loss of knowledge for some tasks. In other way, often, it is not paid attention how well extra tasks are learned because of their only purpose is to help the main task be learned better.


$o^{(m)}$
$o^{(s)}$
Private subnetwork for main task
Common subnetwork for main task
x_1
x_d
x_1
x_d

(a) A MTL scheme with a private subnetwork used by the main task.


$o^{(m)}$
$o^{(s)}$
Private subnetwork for main task
Common subnetwork for main task
x_1
x_d
$t^{(s)}$
x_1
x_d

(b) A MTL scheme with a private subnetwork used by main task and a extra input.

Fig. 3. MTL schemes with a common subnetwork, that learns all tasks, and a private subnetwork, that only learns the main one.

It is possible to improve MTL performance using net schemes more complicated than standard MTL scheme (Figure 2(b)) [15]. One solution is adding a *private* or *specific subnetwork* to learn only the main task. Figure 3(a) shows this scheme. Therefore, there are now two disjoint hidden layers or two disjoint subnetworks. One of them is a private subnetwork used only by the main task, while other is the common subnetwork shared by the main task and the extra task. This common subnetwork supports MTL transfer. This net architecture is asymmetric because the main task can see and affect the private subnetwork used by the extra task, but the extra task can not see or affect the subnetwork reserved for the main task. Up to now, we have supposed that the inputs $\mathbf{x}$ are equal to all hidden neurons, using our notation, $\mathbf{X}^{(s)}$ are the same that $\mathbf{X}^{(m)}$. But we can improve performance of MTL if the desired values $t^{(s)}$ are introduced together with inputs $\mathbf{x}$ as new input-features to learn the main task [17]. Figure 3(b) shows this architecture. Working in this way, we are adding a priori information about domain in private subnetwork and the generalization of main task will be better. An important issue about extra inputs is that, during learning, the targets are known, but not during the operation phase In [17], the concept of *consistency* is used to solve this drawback. It will be also used, and briefly explained, in this work because it is simple and efficient.

V. Proposed Method

In this work, we propose a novel neural network where estimation of missing values is oriented by the learning of the classification task which follows a MTL scheme. Next, we explain how this MTL network learns and works in a general classification problem. After that, training and operation phase with missing data is explained.

A. A MTL Neural Network to Classify Incomplete Input Data

Suppose a c-class classification problem described by N input vectors composed of d real attributes. Consider that m of the d ($m \leq d$) features are incomplete (they have some missing values), where any input vector of these incomplete features can be a missing value, as it is showed in Figure 1(b). Moreover, we define the vector $\mathbf{a} = [a_1, a_2, ..., a_k, ..., a_m]$ whose components are the m incomplete attributes in the data set. Therefore, this problem is composed of two kind of different tasks:

- Main task: one c-class classification task.
- Secondary tasks: m imputation tasks associated to each incomplete feature.

Figure 4 shows a MTL network, based on our proposed method, to solve a general classification problem. In general, each input vector is composed by d units associated with each attribute and, in some cases, c extra inputs associated with the classification target $\mathbf{t}^{(C)}$. There are $m + 1$ subnetworks, one private subnetwork that only learns the classification task, and m common subnetworks where each one of them learns two tasks: the main one and the secondary imputation task associated to each feature with missing data. Each subnetwork can be composed of a different number of hidden neurons. In the output layer, there are $m + c$ outputs distributed in the similar way than inputs: c outputs, $o_1^{(C)}, ..., o_c^{(C)}$, corresponding to classification task and m outputs, $o_1^{(M)}, ..., o_m^{(M)}$, corresponding to the secondary tasks. We use hyperbolic tangent as activation function $g()$ of all hidden neurons and linear outputs in the MTL network showed in Figure 4.

In our notation, $w_{i,j}^{(1)}$ denotes a weight in the first layer, going from input unit i to hidden unit j, and $w_{0,j}^{(1)}$ denotes

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 10, 2009 at 08:25 from IEEE Xplore. Restrictions apply.

the bias for hidden unit j. Notation is similar for weights in the second layer. In all network topologies showed in this work, biases are implicit in order to simplify the figures. With respect to neuron notation, the private subnetwork, that learns the classification task, is labeled with subindex C, and the rest of common subnetworks are labeled with the incomplete feature that they have to learn. For example, the first neuron of the private subnetwork is labeled as 1_C and 3_{a_2} denotes the third neuron of the common subnetwork that learns the a_2 attribute.

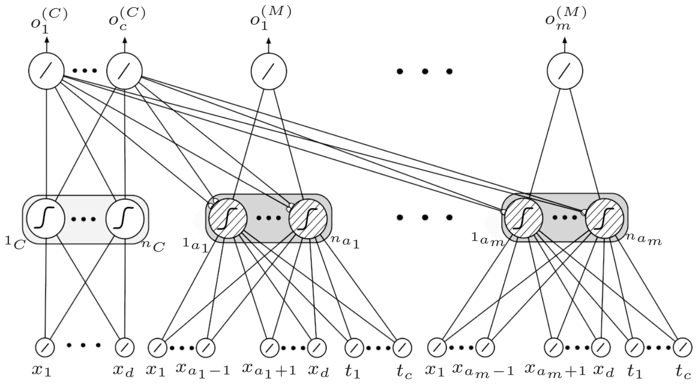


Fig. 4. Proposed MTL neural network that combines classification and imputation. In this network, learning of imputation tasks is oriented by the learning of the classification task. It is composed of $m+1$ subnetworks: one private subnetwork to learn the main classification task (labeled with C), and m common subnetworks for learning the main one and a secondary imputation task (labeled with M) the at same time. Neurons of private subnetworks work as a classical neuron, but neurons of common subnetwork are MTL neurons. Moreover, extra inputs (classification targets) are used in these common subnetworks.

Now, it is explained how neurons process the information depending on the learned tasks and their weight connections. Two kinds of hidden neurons can be considered: classical neuron, that learns only one task, and MTL neuron, that learns at same time several tasks. Neurons of the private subnetwork only learn one task. These neurons work as a classical neuron because they compute the sum product of its weigths and its input signals. Figure 5(a) shows a classical neuron of the private subnetwork whose output can be written as

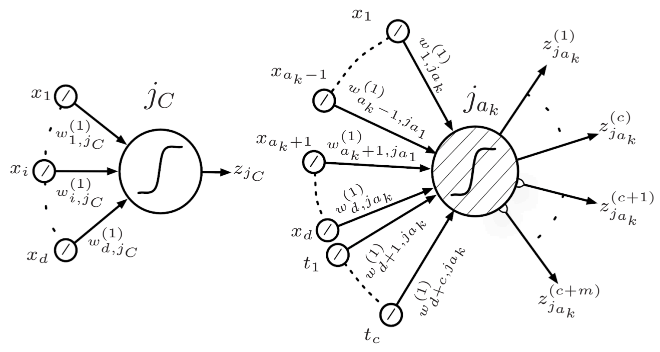
$$z_{j_C} = g\left(\sum_{i=1}^{d} w_{i,j_C}^{(1)} x_i + w_{0,j_C}^{(1)}\right) \quad (2)$$

On the other hand, common subnetworks is composed of neurons that learns at same time all tasks, *i.e.*, they are MTL neurons. We implement them in a different way to a classical neuron. Figure 5(b) shows a MTL neuron in the common subnetwork a_k (which learns the a_k attribute of data, *i.e.*, x_{a_k}). They are connected to all inputs units less that one is associated with, x_{a_k}. To explain it, suppose that, in the common subnetwork a_k, the input x_{a_k} is connected. In that case, there would be a direct connection to map the input as output, and so, imputation output $o_k^{(M)}$ would be dependent of x_{a_k} and the rest of inputs would be omitted (associated weigths would tend to be zero) [15]. For this reason, input-output direct connections are avoided establishing to zero the weights $w_{a_k,j_{a_k}}^{(1)}$, with $k = 1, 2, ..., m$. In Figure 4 and 5(b) are not drawn these weigths. Moreover, classification targets are used as extra inputs only for the secondary tasks Following this, outputs of the MTL neurons compute the following expressions,

$$z_{j_{a_k}}^{(s)} = \begin{cases} g\left(\sum_{i=1}^{d} w_{i,j_{a_k}}^{(1)} x_i + w_{0,j_{a_k}}^{(1)}\right) & s = 1, ..., c \\ g\left(\sum_{i=1}^{d+c} w_{i,j_{a_k}}^{(1)} x_i + w_{0,j_{a_k}}^{(1)}\right) & s = c+1, ..., c+m \end{cases} \quad (3)$$

In Figure 5(b) a representation of a MTL neuron is shown. The two different outputs $z_{j_{a_k}}^{(s)}$, corresponding to the two equations in (3), are marked with a common arrow and an arrow beginning with a circle, respectively. This representation is also used in the Figure 4.

Finally, the outputs of the proposed network are obtained by a linear combination of the outputs of the hidden neurons using a second layer of processing units.



(a) Classical neuron. (b) MTL neuron.

Fig. 5. Types of implemented neurons

Another important issue is the number of neurons in each subnetwork because it determines the complexity of the MTL network. In this paper, we choose a fixed number of neurons in each subnetwork for each tested problem.

Now, total target vector $\mathbf{t}^{(n)}$ is composed by the classification task target vector and the components corresponding to each imputation task, *i.e.*, the attributes with some missing value. Thus, the total target vector for a two dimensional problem with missing values at both attributes can be written as $\mathbf{t}^{(n)} = (t^{(n,\mathrm{C})}, x_{a_1}^{(n)}, x_{a_2}^{(n)})$. Note how the input features with missing values represent the targets of secondary tasks. For example, suppose an input vector $\mathbf{x}^{(3)} = (0.1, 0.25)$ whose desired output $t_3^{(\mathrm{C})}$ is equal to -1, therefore, in the proposed MTL scheme, its total target vector is $\mathbf{t}^{(3)} = (-1, 0.1, 0.25)$.

B. Learning a Classification Task with Missing Inputs

In order to explain how our MTL network works, we divide the proposed scheme in three phases:

1) *Initialization phase*. The weights are initialized and input data set is normalized in this stage.

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 10, 2009 at 08:25 from IEEE Xplore. Restrictions apply.

2) *Learning phase*. The weights are updated and missing data imputation is done.
3) *Operation phase*. It is described the operation of the MTL network once it has been trained.

1) Initialization phase: Before learning, all weights are initialized randomly with values from the interval $\left[-\frac{1}{2}\sqrt{\frac{3}{ninputs}}, +\frac{1}{2}\sqrt{\frac{3}{ninputs}}\right]$, where $ninputs$ is the total number of inputs $d + c$, [18]. Moreover, training set is normalize to zero mean and unit variance, and after that, incomplete values are setting up to zero. This previous zero initialization causes a dynamical adaptation of connections depending on the missing data location in the input vector, because an input equal to zero does not contribute to the learning.

2) Learning phase: Learning is based on the definition of an error function, which is then minimized with respect to the weights (and biases) in the network. In this work, we use the sum-of-squares error function defined as

$$E = \frac{1}{2}\sum_{n=1}^{N}\left(\|\mathbf{o}^{(n,\mathrm{C})} - \mathbf{t}^{(n,\mathrm{C})}\|^2 + \sum_{k=1}^{m}\left(o_k^{(n,\mathrm{M})} - x_{a_k}^{(n)}\right)^2\right) \quad (4)$$

where $\mathbf{o}^{(n,\mathrm{C})}$ and $o_k^{(n,\mathrm{M})}$ are, respectively, the classification output and the k imputation output obtained for the input vector $\mathbf{x}^{(n)}$. We can rewrite (4),

$$E = E^{(\mathrm{C})} + E^{(\mathrm{M})} = E^{(\mathrm{C})} + \sum_{k=1}^{m} E_k^{(\mathrm{M})} \quad (5)$$

where $E^{(\mathrm{C})}$ is the classification error and $E_k^{(\mathrm{M})}$ is the imputation error of the incomplete attribute a_k. These error functions depend on the differences between obtained outputs and targets. If missing values are presented in $\mathbf{x}^{(n)}$, its total target $\mathbf{t}^{(n)}$ will be incomplete. In these cases, it is not possible to compute the differences for incomplete imputation targets because they are unknown. For this reason, differences associated to every incomplete imputation target is established to zero.

After obtaining the differences, the derivatives of the error E with respect to the weights can be evaluated, and these derivatives are used to find weight values which minimize the error function by a gradient optimization method. In order to explain the learning, first, it is necessary to distinguish between weights of output layer, $w_{j,s}^{(2)}$ only are influenced by the learning of the task that they are connected, and weights of input layer, $w_{i,j}^{(1)}$. These weights of the first layer can be divided in two groups: weights associated to a private subnetwork and weights associated to the common subnetwork. The weights of the private subnetwork only are influenced by the error $E^{(\mathrm{C})}$, whereas, the weights of each common subnetwork are influenced both by $E^{(\mathrm{C})}$ and $E_k^{(\mathrm{M})}$ errors. In particular, we use gradient descent method in sequential mode with adaptive learning rate and momentum term.

Another important issue is that incomplete values are estimated using the imputation ouputs during training stage. Learning of the classification task affects to these imputed values, and so, this imputation is oriented to solve the classification task. Imputation is done when the learning of imputation tasks is stopping.

3) Operation phase: The operation of the proposed method depends on the presence of missing values in $\mathbf{x}^{(n)}$. If $\mathbf{x}^{(n)}$ is completely known, imputation is not necessary and the MTL network directly classifies the input pattern using the classification output $\mathbf{o}^{(n,\mathrm{C})}$. But if $\mathbf{x}^{(n)}$ have incomplete data, imputation outputs $o_k^{(n,\mathrm{M})}$ are used to estimate the missing data. These imputation outputs are function of $\mathbf{t}^{(n,\mathrm{C})}$ as part of the input. Nevertheless, this information is not available in the operation mode. In order to solve this problem, we check all possible $\mathbf{t}^{(\mathrm{C})}$ values and the most *consistent* is selected. The consistency of $\mathbf{t}^{(\mathrm{C})}$ is a measure of the difference between $\mathbf{t}^{(\mathrm{C})}$ and the output $\mathbf{o}^{(\mathrm{C})}$ produces by the network after the imputation values of missing data is realized using the corresponding $o_k^{(\mathrm{M})}$.

VI. Experiments and Simulations

In order to test the proposed MTL network introduced in the previous section, three datasets from UCI database are used, [19]. Table I shows these sets. Initially, each dataset is randomly divided into three subsets: 1/3 instances of the dataset are used as training set, 1/6 instances are used as validation set and the rest 1/2 are used as testing set. Such process is repeated ten times and ten groups of training, validation and testing subset are generated. Then, a given percentage of missing data is artificially inserted into all subsets and selected attributes in a completely at random manner. Finally, our method is applied over the training subsets to build up classifiers that are capable to estimate the missing values, and then, these classifiers are used to classify instances in testing subset to obtain the classification accuracy.

TABLE I
Datasets summary

Dataset	Instances	Attributes	Classes
Iris Plant	150	4	3
Glass	214	9	6
Pima Indians	768	8	2

A. The Iris Plant Problem

In Iris Plant problem, the goal is to classify irises based on four attributes: sepal length (A1), sepal width (A2), petal length (A3) and petal width (A4). This problem has been used widely in many works, and the classification error without deleting data is around $3\% \sim 4\%$, [19].

In particular, we insert missing values randomly in all possible combinations of the four attributes for different percentage of missing data. We have done this both to evaluate the influence of the missing data in each one of

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 10, 2009 at 08:25 from IEEE Xplore. Restrictions apply.

TABLE II

MUTUAL INFORMATION BETWEEN EACH ATTRIBUTE AND THE CLASSIFICATION TASK OF THE IRIS PROBLEM.

	Attributes			
	A1	A2	A3	A4
MI	0.877	0.511	1.446	1.436

the features, and to check how the different combinations of incomplete attributes affects to the learning and the classification accuracy. It is clear that not all the attributes are equally important to classification task. We measure this importance with the Mutual Information (MI) between each attribute and the classification task [13]. Table II shows the MI for each attribute of the Iris problem, and Table III summarizes the obtained results for each possible combination of incomplete features with percentages of missing data equal to 30% and 40%. The first column of this table indicates which attributes are incomplete, labeled with a 1, or complete, labeled with a 0.

TABLE III

OBTAINED MISCLASSIFICATION RATES FOR IRIS PROBLEM.

Attributes				**Missing Rate**	**Training (%)**	**Test (%)**
A1	A2	A3	A4		Mean ± SD	Mean ± SD
1	0	0	0	30%	2.00 ± 0.10	4.00 ± 0.60
1	0	0	0	40%	2.20 ± 1.00	4.53 ± 0.88
0	1	0	0	30%	2.40 ± 0.80	3.20 ± 0.88
0	1	0	0	40%	2.60 ± 1.00	3.33 ± 0.67
0	0	1	0	30%	2.60 ± 1.20	5.07 ± 0.53
0	0	1	0	40%	1.80 ± 1.08	4.80 ± 0.88
0	0	0	1	30%	1.80 ± 1.08	4.13 ± 1.11
0	0	0	1	40%	3.00 ± 1.34	4.40 ± 1.04
1	1	0	0	30%	2.40 ± 1.49	3.33 ± 0.67
1	1	0	0	40%	2.80 ± 1.33	3.73 ± 1.53
1	0	1	0	30%	2.40 ± 1.50	5.07 ± 0.53
1	0	1	0	40%	2.00 ± 1.26	5.60 ± 0.80
1	0	0	1	30%	2.00 ± 0.89	4.40 ± 1.20
1	0	0	1	40%	3.00 ± 1.84	5.60 ± 1.16
0	1	1	0	30%	3.20 ± 1.33	4.40 ± 0.85
0	1	1	0	40%	3.40 ± 2.01	4.27 ± 0.53
0	1	0	1	30%	2.80 ± 1.33	3.07 ± 0.61
0	1	0	1	40%	2.60 ± 1.28	3.73 ± 0.10
0	0	1	1	30%	3.60 ± 3.32	9.07 ± 1.55
0	0	1	1	40%	4.20 ± 3.28	11.07 ± 1.89
1	1	1	0	30%	3.20 ± 2.04	4.67 ± 0.89
1	1	1	0	40%	4.00 ± 1.26	4.93 ± 0.85
1	1	0	1	30%	2.20 ± 1.44	3.33 ± 0.67
1	1	0	1	40%	2.60 ± 1.35	4.33 ± 0.67
1	0	1	1	30%	6.00 ± 2.57	10.40 ± 2.05
1	0	1	1	40%	4.00 ± 3.22	16.93 ± 2.92
0	1	1	1	30%	6.60 ± 5.87	10.40 ± 1.55
0	1	1	1	40%	6.80 ± 4.75	12.40 ± 2.54
1	1	1	1	30%	3.60 ± 1.96	10.13 ± 2.00
1	1	1	1	40%	3.00 ± 2.41	16.67 ± 1.81

When attributes with higher values of MI are incomplete (attributes A3 and A4), obtained results are worse than those obtained when missing values are in attributes with lower MI (attributes A1 and A2). Another important issue is that unrelated attributes are less influenced by the learning of the classification task than more related (cases A3 and A4). To show it, Figure 6 illustrates the evolution of sum-of-squares error for each task when there is a 10% of missing data in the all attributes. We can see how the secondary tasks associated to the attributes A3 and A4 are learned easier and better than the rest of the extra tasks. In this problem, imputation is done when the learning of the secondary tasks is stopping. In the epoch 27, where the first imputation is done, the training error associated to the main task decreases in a sudden way. Each one of the following imputations affects less gradually in the learning of the classification task because of the learning of the secondary tasks is stooped gradually.

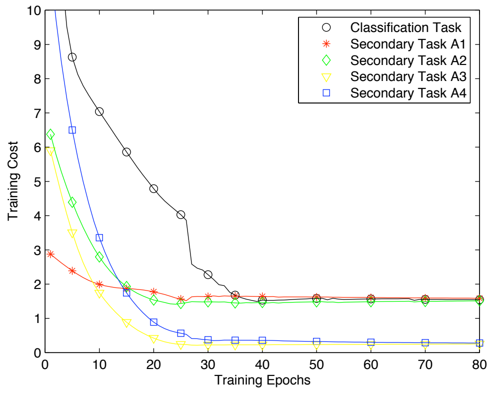


Fig. 6. Evolution of the sum-of-squares error during learning for each task in the Iris problem with 10% of missing data in all attributes.

B. Forensic Glass Problem

This data set contains the description of 214 fragments of glass originally collected for a study in the context of criminal investigation. Each instance is composed of nine attributes, labeled as A1, A2, ..., A9. Classification error without deleting data is around 35% using MLP, [19]. We measure the attribute's importance and its relation with the classification task using the MI. In order to test our method, we insert incomplete values randomly for different percentage of missing data in the two most related features (A1 and A2) and in two least related attributes (A8 and A9).

As we can see in Figure 7, the task associated with the least relevant attributes are learnt not as well as the task associated with A1 and A2 attributes. Table IV summarizes the obtained results in this problem for different missing data rate. In this problem, proposed method obtains a similar accuracy than in the classification using the complete data set.

C. Pima Indians Diabetes Problem

Pima Indians Diabetes data set was originally collected on a population of women in order to diagnose diabetes using eight attributes (A1, A2, ..., A8). It can be found in the web page of B. D. Ripley's book [20]. In this case, there are 3 different sets. One of them is for test and consists of 332

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 10, 2009 at 08:25 from IEEE Xplore. Restrictions apply.

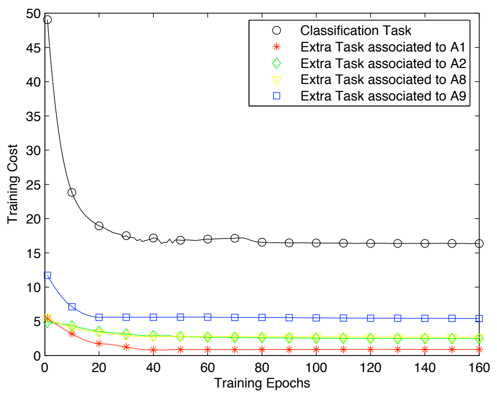


Fig. 7. Evolution of the sum-of-squares error during learning for each task in the Glass problem with 10% of missing data.

TABLE IV

OBTAINED MISCLASSIFICATION RATES FOR GLASS PROBLEM.

Missing Rate	**Training (%)**	**Test (%)**
	Mean $\pm$ SD	Mean $\pm$ SD
10%	18.10 ± 4.35	31.50 ± 3.91
20%	16.70 ± 3.44	33.33 ± 1.67
30%	19.00 ± 4.84	30.67 ± 2.49
40%	18.20 ± 2.04	35.67 ± 3.35

complete cases. Two remaining sets are for training: one has only 200 complete cases, and the other one has 200 complete cases and 100 incomplete cases. In particular, three different attributes presents missing data: the attributes A3, A4 and A5. Table 3 shows the MI between them and the classification task, and also, it shows the percentages of missing data in each attribute. As we can see in this table, the attribute A5 is the most related to the classification task.

TABLE V

PERCENTAGES OF MISSING DATA, MUTUAL INFORMATION BETWEEN EACH INCOMPLETE ATTRIBUTE AND THE CLASSIFICATION TASK FOR THE PIMA INDIANS PROBLEM.

	Attributes		
	A3	A4	A5
Missing Rate	4.33%	32.67%	1.00%
MI	0.111	0.232	0.534

Figure 8 show the evolution of the training cost for each task during the learning. As we can see in this figure, the task associated to A3 presents a worse learning than the rest ones. It is due to that its MI value is the smallest one and therefore it is the least related attribute with the main task. On the other hand, tasks associated to attributes A4 and A5 use the advantages of the common learning with the main task because they are more related than the attribute A3. Another issue is that the estimated values for missing data do not contribute as clearly to the learning of the classification task as in the Iris problem. Nevertheless, obtained results are better than when only complete cases are used, as we can see next. Misclassification rates in test for the Pima Indians database are the following: 23.34 ± 1.63 % with only complete cases, and 19.92 ± 0.59 % using the proposed MTL network. The obtained training set, composed by complete cases and incomplete cases with imputed values, produces a better generalization, i.e., imputed values are those that contribute to improve the main task learning.

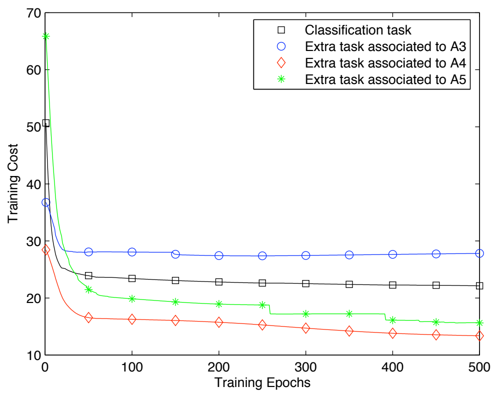


Fig. 8. Evolution of the sum-of-squares error for each task during learning in the Pima Indians problem.

VII. CONCLUSIONS AND FUTURE WORKS

In this work, we have established a neural network to classify incomplete input vectors with numerical attributes and estimate the missing values using the advantages of MTL. Unlike other proposed methods, classification and missing data estimation are combined in only one neural network using subnetworks. To implement it, we have used the classification as main task and each incomplete feature as a secondary task. Each one of them has associated a common subnetwork that learns at the same time the secondary task and the main one. There is also a private subnetwork that learns specifically the main task. Weights connections are dynamically adapted in function of the missing attributes for every input vector, independently of how missing data are distributed, and moreover, we use the outputs that learn incomplete features to estimate missing values during learning process. Doing this, classification task helps to learn these secondary tasks, *i.e.*, classification task guides the imputation process during the learning of all tasks in parallel, and the secondary tasks help to improve the generalization capabilities of the main task. Moreover, imputed values are those that contribute to get a better generalization because the learning of imputation tasks is oriented by the learning of the main task; but classification accuracy is the fundamental aim and not how good the imputed values are. Another important improvement is obtained when classification targets are used as extra inputs in the subnetworks associated to extra tasks. During the operation phase, the most consistent class is chosen. Experimental results for artificial and real incomplete

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 10, 2009 at 08:25 from IEEE Xplore. Restrictions apply.

databases proved these arguments.

This work will stimulate future works in many directions. Some of them are using different error functions (cross-entropy error in discrete tasks, and sum-of-squares error in continuous tasks), adding an EM-model to probability density estimation into the proposed MTL scheme, setting the number of neurons in each subnetwork dynamically using constructive learning, an extensive comparison with other imputation methods, to use this procedure in regression problems, and extending the proposed method to different machines, e.g., Support Vector Machines (SVM).

REFERENCES

[1] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data*, 2nd ed. New Jersey, USA: John Wiley & Sons, 2002.

[2] J. L. Schafer, *Analysis of Incomplete Multivariate Data*, 1st ed. Florida, USA: Chapman & Hall, 1997.

[3] R. Caruana, "Multitask learning: a knowledge-based source of inductive bias", *Proceedings of the 10th International Conference of Cognitive Science*, pp. 41-48, 1993.

[4] J. Baxter, *Learning internal representations*, Ph. D. Thesis, Flinders University of South Australia, Adelaide, 1994.

[5] C. M. Bishop, *Neural networks for pattern recognition*. New York, USA: Oxford University Press, 1995.

[6] S. Nordbotten, "Neural network imputation applied to the Norwegian 1990 Census Data", *Journal of Official Statistics*, vol. 12, no. 4, pp. 385-401. Netherlands: Kluwer, 1996.

[7] S. Y. Yoon and S. Y. Lee, "Training algorithm with incomplete data for feed-forward neural networks", *Neural Processing Letters*, no. 10, pp. 171-179. Netherlands: Kluwer, 1999.

[8] M. K. Markey, G. D. Tourassi, M. Margolis and D. M. DeLong, "Impact of missing data in evaluating artificial neural networks trained on complete data", *Computers in Biology and Medicine*. (accepted paper, in press).

[9] H. Ishibuchi, A. Miyazaki, K. Kwon and H. Tanaka, "Learning from incomplete training data with missing values and medical application", *Proceedings of 1993 International Joint Conference on Neural Networks*, pp. 1871-1874. Nagoya, Japan, 1994.

[10] H. Ishibuchi, A. Miyazaki and H. Tanaka, "Neural-network-based diagnosis systems for incomplete data with missing inputs", *Proceedings of IEEE World Congress on Computational Intelligence*, vol. 6, pp. 3457-3460. Orlando, USA, 1994.

[11] Zs. J. Viharos, K. Novaki and T. Vincze, "Training and application of artificial neural networks with incomplete data", *Proccedings of* 15^{th} *International Conference on Industrial & Engineering Applications of Artificial Intelligence & Experts Systems*, LNCS, pp. 64-659. Cairns, Australia, 2002.

[12] S. Thrun, "Is learning the n-thing any easier than learning the first?", *Advances in Neural Information Processing Systems (NIPS)*, pp. 640-646, 1996

[13] D. Silver, *Selective Transfer of Neural Network Task Knowledge*, Ph.D. Thesis, University of Western Ontario, 2000.

[14] D. Silver and R. Mercer. "Selective Functional Transfer: Inductive Bias from Related Tasks", *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2001)*, pp. 182-189. Cancun, Mexico, ACTA Press. 2001.

[15] R. Caruana, *Multitask learning*. Ph. D. Thesis. Carnegie Mellon University. 1997.

[16] J. Ghosn and Y. Bengio, "Bias Learning, Knowledge Sharing", *IEEE Transactions on Neural Networks*, vol. 14, no. 4, pp. 748-765. 2003.

[17] P. J. García-Laencina, A. R. Figueiras-Vidal, J. Serrano-García, J. L. Sancho-Gómez, "Exploiting multitask learning schemes using private subnetworks", *Proceedings of the 8th International Work-Conference on Artificial Neural Networks*, pp. 233-240, Barcelona, Spain, 2005.

[18] L. Bottou and P. Gallinari, "A Framework for the Cooperation of Learning Algorithms". Technical Report, Laboratorie de Recherche en Informatique, Universite de Paris XI, 9145 Orsay Cedex, France, 1991.

[19] C. J. Merz and P. M. Murphy, UCI Repository of Machine Learning Datasets, 1998. http://www.ics.uci.edu/ mlearn/MLRepository.html

[20] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, 1996. http://www.stats.ox.ac.uk/pub/PRNN/

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 10, 2009 at 08:25 from IEEE Xplore. Restrictions apply.