

# Values Deletion to Improve Deep Imputation Processes

Adrián Sánchez-Morales<sup>1</sup>(✉), José-Luis Sancho-Gómez<sup>1</sup>,  
and Anibal R. Figueiras-Vidal<sup>2</sup>

<sup>1</sup> Department of Information and Communications Technologies,  
Universidad Politécnica de Cartagena, Plaza del Hospital,  
1, 30202 Cartagena (Murcia), Spain  
adrian.sm91@gmail.com

<sup>2</sup> Signal Theory and Communications Department,  
Universidad Carlos III de Madrid, Madrid, Spain

**Abstract.** Most machine learning algorithms are based on the assumption that available data are completely known, nevertheless, real world data sets are often incomplete. For this reason, the ability of handling missing values has become a fundamental requirement for statistical pattern recognition. In this article, a new proposal to impute missing values with deep networks is analyzed. Besides the real missing values, the method introduces a percentage of artificial missing ('deleted values') using the true values as targets. Empirical results over several UCI repository datasets show that this method is able to improve the final imputed values obtained by other procedures used as pre-imputation.

## 1 Introduction

Nowadays, regardless of the industry you work in, data plays an important role. However, it has been shown through the literature the wide range of drawbacks which can appear when one faces real applications. One of the most common is the presence of unknown data so that data processing is a necessity. Otherwise, missing values may cause false results on the main problem [1].

Trying to find out the best method to handle missing attributes in a database is another issue studied by many authors [2,3], and it depends in most cases on the kind of missing is being dealt with. Although there exist several ways of handling this problem, imputation has become one of the most studied due to missing treatment is independent of the learning algorithm used in a following stage. This technique consists of filling unknown values with estimated ones, and there is a wide family of this methods, from simple imputation techniques like mean substitution to those which analyze the relationships between attributes like support vector machines [2].

In this paper, we will consider the high representation capacity of Autoencoders (AE) to create an efficient method of imputation based on a deep architecture. In particular, Denoising Autoencoders (DAEs) will be used to form a Stacked Denoising Autoencoder (SDAE) through a layer-wise procedure [4].

Moreover, this imputation method is able to leverage all the available information in a data set, including that which is in incomplete instances. We will show how our method is able to improve the final imputed values obtained by other procedures used as pre-imputation and that in general give good results. We use for that some of the UCI repository data sets where different amount of missing values will be artificially inserted in order to compute the error between real and final imputed values.

The remainder of this paper is structured as follows: in Sect. 2 details of the proposed deep learning technique are given and the learning algorithm is presented. Experiments of applying the proposed method to solve some synthetic problems and a review of the used pre-imputation techniques are shown in Sect. 3. Finally, the paper is closed by the conclusions and future related works in Sect. 5.

## 2 Proposed Method

As we have already mentioned above, an imputation method based on deep architectures is presented in this paper. Researchers have demonstrated how in a well-trained deep neural network, the hidden layers learn a good representation of the input data [5–7], and it is also widely known the capability of reconstruction of DAEs. If we review the concept of *denoising*, a neural network is constructed to provide ‘clean’ outputs from noisy inputs, which are obtained adding noise to the original patterns [4]. Learning is done through unsupervised training where the “clean” inputs are used as target.

These concepts will be used in order to carry out an efficient imputation procedure. Therefore, as a first approach, a SDAE will be created by stacking DAEs; then, it will be checked if the results can be improved by applying a new technique based on deleting samples.

### 2.1 SDAE Imputation Method

In most machine learning imputation techniques, the model is trained only using complete instances what implies a lost of information that could be critical in the problem. However, this issue will be overcome thanks to a simple modification of the Stochastic Gradient Descent (SGD) algorithm when it is used to train a SDAE.

Let us consider an unsupervised training data set with the form  $\{\mathbf{x}_n, \mathbf{m}_n\}_{n=1}^N$ ,  $n = 1, \dots, N$ , where  $\mathbf{x}_n$  is the  $n$ -th pattern and  $\mathbf{m}_n$  is a vector with the same dimension which indicates whether  $\mathbf{x}_n$  presents missing values through:

$$m_{nd} = \begin{cases} 1, & x_{nd} \text{ is a known value,} \\ 0, & x_{nd} \text{ is missing,} \end{cases} \quad (1)$$

where  $d$  ( $1 \leq d \leq D$ ) denotes component. These missing values are firstly filled in by one of the pre-imputation methods we will present in next section. Then, a SDAE is trained to reconstruct a noisy version of the input by minimizing

a *Mean Squared Error* (MSE) function and overlooking error corresponding to missing values. This noise is inserted in every sample except those with missing, and weights in networks are iteratively updated to minimize

$$E = \sum_{n=1}^N \| (\mathbf{z}_n - \mathbf{x}_n) \odot \mathbf{m}_n \|^2 \quad (2)$$

where  $\mathbf{z}_n$  is the output of the SDAE which depends on the weights of the network and  $\odot$  represents the direct product, also called Hadamard product.

Therefore, through an unsupervised layer-wise training and a subsequent fine tuning, a deep DAE is constructed by stacking DAEs. The training process to construct SDAEs is the same as that followed in [4]. This neural network is able to reproduce the input in its output layer, taking into account not only the complete instances but also the known characteristics of the incomplete patterns. The idea is to impute unknown values at the same time as it is learnt to denoise input data.

## 2.2 Deleting Values

In the approach mentioned above, the SDAE is trained to make predictions for missing values and reconstruct noisy versions of the input patterns.

Deleting some input values (values in some features of the input patterns) is now proposed so that the network can learn the true ones, i.e., some known input values are deleted—they can be considered as an artificial missing values—being these known values used as targets. In this way, a deep neural network can be trained to reconstruct noisy inputs, predicts both ‘real’ missing values and “deleted” values. Thus, due to the fact that deleted values are treated as missing but its targets are known, real missing values can be predicted more accurately.

Let us suppose we face a data problem with missing values in a particular characteristic (the extension to more characteristics is direct). The idea described above is to suppress a percentage of known values of that characteristic, playing with the advantage that the actual values are known and can be used to guide learning by using them as targets. Thus, if we define  $\mu$  as the percentage of missing values and  $\varepsilon$  as the percentage of deleted samples, we will have  $(\mu + \varepsilon)N$  samples with unknown values in the training set while there are only  $\mu N$  missing samples in the test set.

## 3 Experiments

The main objective of the experiment conducted in this work is to see how the imputation of missing values provided by several well-known methods can be improved by means of our proposed algorithm based on SDAEs.

Three datasets are selected from UCI repository [8] corresponding to classification and regression problems. In the UCI database, these sets are referred to as Cloud Dataset, Blood Transfusion Service Center and Boston Housing.

**Table 1.** Datasets.

	Samples		
	Total	Classes	Features
Cloud dataset	1024	-	10
Blood transfusion	683	2	4
Boston housing	506	-	13

While the descriptive features of these sets are shown in Table 1, a more detailed description of the sets can be found in [8].

Only complete datasets are considered to artificially insert a variety of missing values in order to compute the error between true and imputed values. Moreover, missing values will be inserted in continuous features.

Every architecture in this survey has been trained according to the techniques mentioned in the previous section. In all cases, SDAEs are constructed to have three hidden layers with 25%–75% of expansion and the number of hidden nodes is set by executing a 5-fold cross validation process applied with 50 different training runs (i.e., different network weights initialization). In order to do that, every data set is split into training (80%) and test (20%). During experiments, several missing percentages  $\mu$  and different amounts of deleted samples  $\varepsilon$  are treated to test the performance of our strategy within different situations. More specifically, values of 10%, 20% and 30% of missing are explored while a percentage from 25% to 100% of the rest of samples in the same characteristic is deleted. Thus, for each missing percentage introduced, 25%, 50%, 75% and 100% of deleted samples are explored.

### 3.1 Pre-imputation Methods

Some of the well-known methods that have been widely studied in the literature are selected to be implemented in the pre-imputation stage. To demonstrate whether the capacity of the pre-imputation technique is key to getting better the final result, three methods with different levels of complexity are used. These are as follows:

- *Zero Imputation (Z)*. A simple and general imputation procedure which consist of initializing unknown values to zero. This is perhaps the simplest method and one of the most carried out in practice, even though it does not take into account the correlation structure of the data. For this reason, it tends to yield poor imputation results.
- *K-Nearest Neighbour Imputation (K)*. The  $K$  method is an imputation procedure based on the principle that the instances with similar properties in a data set are close to each others. This mechanism is one of the most popular approaches for solving incomplete data problems. Basically, given an unknown value, its  $K$  nearest neighbours are obtained among the training cases with

known values in the attribute to be imputed. Then, the missing value is filled in with an estimation calculated by using those  $K$  values. The optimal value of  $K$  is computed by cross validation in this work.

- *Support Vector Machine Imputation (S)*. According to this technique, a SVM can be used to impute missing values as regression analysis. A different SVM is used to learn every feature with missing values. Therefore, every complete training example  $\mathbf{x}$  is used to train the network in the following way. Without loss of generality, suppose we have a data set with missing values only in the  $j$ -th feature. Then the input pattern to train the SVM will be composed of  $x_1, x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_D$ , and the target value by  $x_j$ . After the network is trained, it is used to perform regression on the instances which have unknown values.

### 3.2 Results

For each data set, the MSE between real and imputed values is shown in a table where the three pre-imputation methods mentioned above are distinguished. Missing is introduced in one feature randomly selected between the continuous features. Results show mean and standard deviation for 50 runs. As it has been said before, values within  $\{0.1, 0.2, 0.3\}$  are explored for  $\mu$  and  $\{0.25, 0.5, 0.75, 1\}$  for  $\varepsilon$ . In addition, best result of a family of pre-imputations is shown in italic and the global minimum for a missing percentage in bold (in some cases, there is more than one bold value for each percentage (columns) because there is no statistical difference between them).

**Cloud Dataset.** In this set, both missing and deleted samples are inserted in the feature 7. Results can be seen in Table 2.

**Table 2.** Results for Cloud dataset.

Procedure	$\mu$		
	0.1	0.2	0.3
$Z$	$1.19 \pm 0.17$	$0.98 \pm 0.25$	$1.09 \pm 0.17$
$Z$ -SDAE	$0.35 \pm 0.03$	$0.42 \pm 0.06$	$0.4 \pm 0.02$
$Z$ -SDAE ( $\varepsilon$ )	<i><math>0.08 \pm 0.04</math> (0.25)</i>	<i><math>0.04 \pm 0.02</math> (0.75)</i>	<i><math>0.07 \pm 0.03</math> (1)</i>
$K$	$0.20 \pm 0.02$	$0.37 \pm 0.01$	$0.51 \pm 0.01$
$K$ -SDAE	$0.08 \pm 0.01$	$0.12 \pm 0.007$	$0.17 \pm 0.008$
$K$ -SDAE ( $\varepsilon$ )	<i><math>0.02 \pm 0.01</math> (0.75)</i>	<b><math>0.02 \pm 0.01</math> (0.5)</b>	<b><math>0.02 \pm 0.007</math> (0.75)</b>
$S$	$0.14 \pm 0.01$	$0.29 \pm 0.012$	$0.31 \pm 0.016$
$S$ -SDAE	$0.06 \pm 0.02$	$0.14 \pm 0.015$	$0.13 \pm 0.01$
$S$ -SDAE ( $\varepsilon$ )	<b><math>0.01 \pm 0.009</math> (0.75)</b>	<b><math>0.02 \pm 0.007</math> (0.5)</b>	<b><math>0.02 \pm 0.015</math> (1)</b>

**Blood Transfusion Service Center.** In this case, missing and deleting is inserted in characteristic number 2. Results are shown in Table 3.

**Table 3.** Results for Blood Transfusion dataset.

Procedure	$\mu$		
	0.1	0.2	0.3
<i>Z</i>	0.81 ± 0.08	0.8 ± 0.05	0.81 ± 0.05
<i>Z</i> -SDAE	0.54 ± 0.09	0.46 ± 0.09	0.69 ± 0.15
<i>Z</i> -SDAE ( $\varepsilon$ )	<i>0.04 ± 0.03 (0.5)</i>	<i>0.07 ± 0.015 (1)</i>	<i>0.04 ± 0.016 (0.75)</i>
<i>K</i>	0.03 ± 0.025	0.04 ± 0.012	0.04 ± 0.01
<i>K</i> -SDAE	0.012 ± 0.006	0.025 ± 0.01	0.02 ± 0.006
<i>K</i> -SDAE ( $\varepsilon$ )	<b>0.005 ± 0.002 (1)</b>	<b>0.006 ± 0.004 (1)</b>	<i>0.006 ± 0.003 (1)</i>
<i>S</i>	0.0158 ± 0.01	0.02 ± 0.005	0.026 ± 0.01
<i>S</i> -SDAE	0.012 ± 0.005	0.016 ± 0.01	0.016 ± 0.005
<i>S</i> -SDAE ( $\varepsilon$ )	<b>0.004 ± 0.003 (1)</b>	<b>0.005 ± 0.002 (1)</b>	<b>0.004 ± 0.001 (1)</b>

**Boston Housing.** For this dataset, different values for  $\mu$  and  $\varepsilon$  are inserted in feature number 13, and results are presented in Table 4.

**Table 4.** Results for Boston Housing dataset.

Procedure	$\mu$		
	0.1	0.2	0.3
<i>Z</i>	1.41 ± 0.05	1.09 ± 0.03	1.15 ± 0.03
<i>Z</i> -SDAE	0.83 ± 0.03	0.93 ± 0.05	0.8 ± 0.02
<i>Z</i> -SDAE ( $\varepsilon$ )	<i>0.69 ± 0.04 (1)</i>	<i>0.62 ± 0.03 (1)</i>	<i>0.7 ± 0.01 (1)</i>
<i>K</i>	0.74 ± 0.05	0.69 ± 0.03	0.78 ± 0.02
<i>K</i> -SDAE	0.58 ± 0.04	0.45 ± 0.02	0.48 ± 0.01
<i>K</i> -SDAE ( $\varepsilon$ )	<b>0.46 ± 0.02 (0.75)</b>	<b>0.43 ± 0.02 (0.75)</b>	<b>0.39 ± 0.015 (1)</b>
<i>S</i>	0.61 ± 0.04	0.54 ± 0.016	0.64 ± 0.016
<i>S</i> -SDAE	<b>0.45 ± 0.018</b>	0.49 ± 0.014	<i>0.43 ± 0.016</i>
<i>S</i> -SDAE ( $\varepsilon$ )	<b>0.46 ± 0.03 (1)</b>	<b>0.45 ± 0.02 (0.75)</b>	0.51 ± 0.016 (1)

## 4 Discussion

From the results shown in the tables, it is observed that for each family of algorithms (*Z*, *K* and *S*), the best imputation result (in italic) is always obtained when deleting samples is introduced ( $\varepsilon \neq 0$ ). There is only one case where this

does not occur, in the Boston Housing with  $\mu = 0.3$  where the best result is obtained with S-SDAE procedure. In addition, the best overall result (in bold) is always obtained by deleting samples and with high values of  $\varepsilon$  (mostly 0.75 and 1). This allow us to infer that the elimination of samples makes the machine able to learn a reconstruction of the missing values with greater precision.

It can also be observed that the method of imputation used is decisive for the final results: the better the method of pre-imputation, the better results are obtained.

## 5 Conclusions

In this paper, a new deep learning-based imputation technique has been presented. Although the unsupervised layer-wise training with Autoencoders (AEs) has been widely studied through the literature, we have carried out a method which allows a better imputation by stacking several Denoising AEs (DAEs). The proposed imputation method is able to leverage all the available information in a data set, including that which is in incomplete instances. Moreover, by means of inserting artificial missing values ('deleted' values), the quality of the imputed values is improved. Empirical results over several UCI repository datasets show that our method is able to improve the final imputed values obtained by other procedures used as pre-imputation.

## References

1. García-Laencina, P.J., Sancho-Gómez, J.L., Figueiras-Vidal, A.: Pattern classification with missing data: a review. *Neural Comput. Appl.* **9**(1), 1–12 (2009)
2. Batista, G., Monard, M.C.: An analysis of four missing data treatment methods for supervised learning. *Appl. Artif. Intell.* **17**, 519–533 (2003)
3. Luengo, J., García, S., Herrera, F.: On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowl. Inf. Syst.* **32**(1), 77–108 (2012)
4. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **11**(11), 3371–3408 (2010)
5. Bengio, Y.: Learning deep architectures for AI. Technical report, Dept. IRO, Université de Montréal (2009)
6. Deng, L., Yu, D.: Deep learning: methods and applications. *Found. Trends Sig. Process.* **7**(3–4), 197–387 (2014)
7. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. *Adv. Neural Inf. Process. Syst.* **19**, 153 (2007). (NIPS06), (B. Scholkopf, J. Platt, and T. Hoffman, eds.)
8. Lichman, M.: UCI machine learning repository (2013). <http://archive.ics.uci.edu/ml>