ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7. Available from http://www.i6doc.com/en/.

A Multi-ELM Model for Incomplete Data

Baichuan Chi[1], Amaury Lendasse[1], Edward Ratner[2] and Renjie Hu[1] *

1- ILT Department, University of Houston, Houston, USA
2- Edammo Inc. Iowa City, USA

Abstract. This paper presents a novel model of Extreme Learning Machines (ELMs) for incomplete data. ELMs are fast accurate randomized neural networks. Nevertheless ELM can only be applied on the complete dataset. Therefore, a novel Multi-ELM Model for incomplete data is proposed, consisting of multiple secondary ELMs and one primary ELM. The secondary ELMs are approximating the primary ELM's hidden neurons' outputs for the data with missing values. As summarized in the experimental Section, this model can be applied on data with any missing patterns, without using imputations and can outperform the traditional imputation methods within a reasonable fraction of missing values, as it avoids the noises introduced by imputations.

1 Introduction

Datasets with missing values (incomplete data) are observed in many real-world machine learning tasks. Values are missing because of human errors, device malfunction, missing intentionally and so on [1]. The commonly used strategies for dealing with incomplete data include discarding the incomplete data and creating imputations of the incomplete data[2]. Both strategies can work well when the fraction of missing values is small[3]. When a considerable amount of data is missing, simply discarding the incomplete data becomes impractical, and imputation tends to introduce more noises and error to the data[4, 5].

The most straightforward imputation is the Mean imputation (MI), in which the mean of the observed values for each variable is computed and the missing values for that variable are imputed by this mean. MI is a very popular method when only a few samples contain missing values. However MI can introduce severely biased estimates (see [6, 7]).

Another widely used imputation method is the K Nearest Neighbors (KNN) imputation [8], which searches for the k closest neighbors to the incomplete observations based on the known values and then impute the missing values based on the non-missing values in the neighbors. KNN imputation can be effective when only a few values are missing [8], but is ineffective when multiple components are missing for one data sample.

Besides imputation and discarding the incomplete data, the third approach is to apply the machine learning models directly on the incomplete data to avoid the errors introduced by imputations, however, not many machine learning models can handle incomplete data directly[3]. This paper proposes a novel machine learning model based on the Extreme Learning Machines [9, 10], which could

*Corresponding author, Email: rhu7@uh.edu be applied directly on the incomplete data, without performing any imputation. This proposed method is capable of handling a variety of missing patterns in the data, and has been proved to be more efficient when the percentage of missing values is high in data.

In the Section 2 the detailed methodology of this proposed method is explained. Next in Section 3 the experiments have been conducted to test the performance of the proposed method. Conclusions are drawn in Section 4.

2 Methodology

2.1 Extreme Learning Machines

The Extreme Learning Machines [11, 12] are Single-Layer Feed-forward Networks (SLFNs) [13]. According to Huang et al. in [11], ELM has universal approximation capability , ELM can approximate any continuous target function with adjustable hidden nodes and produce good generalized performance with extremely fast learning process.

ELM contains three layers of neurons. In ELM the input layer weights ($\boldsymbol{w}$) and biases ($\boldsymbol{b}$) are generated randomly and stay unchanged afterwards. These input weights map the input data $\boldsymbol{x}$ to a higher-dimensional feature space. A nonlinear transformation is followed after the mapping process and produce the hidden layer output as: $h_i(\boldsymbol{x}) = \phi(\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{w_i} + b_i)$, $i \in [1, L]$, where ϕ is called an activation function (see [14]), L is the number of neurons in the ELM's hidden layer. This hidden layer output serves as the input of an ordinary least square (OLS) problem: $\mathbf{T} = \sum_{i=1}^{L} \beta_i h_i(\boldsymbol{x})$, where β is the weight to be solved in the OLS problem, and $\mathbf{T}$ is the target variable or the dependent variable.

To solve the above OLS problem in ELM for a set of N distinct pair of observations $(\boldsymbol{x}_j, t_j)$, $j \in [1, N]$, with $\boldsymbol{x}_j \in \mathbb{R}^d$ being the input, and t_j being the corresponding target, the matrix form of h_i is introduced as $\mathbf{H}$ and the OLS problem is represented as: $\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta}}{\arg\min} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2^2$, and

$$\mathbf{H} \quad = \quad \begin{pmatrix} h_1(\boldsymbol{x}_1) & \dots & h_L(\boldsymbol{x}_1) \\ \dots & \ddots & \dots \\ h_1(\boldsymbol{x}_N) & \dots & h_L(\boldsymbol{x}_N) \end{pmatrix}. \tag{1}$$

The final solution $\boldsymbol{\beta}^*$ is solved as $\boldsymbol{\beta}^* = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}$. With the optimal OLS weights, approximation of the target value $\hat{t}$, and the ELM function $\mathfrak{E}(\boldsymbol{x})$ can be written as the following equation. $\hat{t} = \mathfrak{E}(\boldsymbol{x}) = \sum_{i=1}^{L} \beta_i^* h_i(\boldsymbol{x})$.

2.2 A Multi-ELM Model for Incomplete Data

2.2.1 Notations for Incomplete Data

In a dataset $\boldsymbol{X}$, a data point $\boldsymbol{x} = (x^1, x^2, \dots, x^d)^T$ is a complete data point if none of the d variables are missing; otherwise if any of the d variables are missing it is an incomplete data point. Typically in an incomplete dataset, there is a portion of complete data, which is denoted as $\boldsymbol{X}_C$. The incomplete data subset is denoted as $\boldsymbol{X}_M$.

Depending on how the variables are missing, each of the incomplete data point $\boldsymbol{x} \in \boldsymbol{X}_M$ has a unique missing pattern $\boldsymbol{p}_j = (p^1, p^2, \dots, p^d)$, where $p^i = 1$ means the ith variable is missing, $p^i = 0$ means variable i is known. $\boldsymbol{x}$ can be reorganized according to it's missing pattern $\boldsymbol{p}_j$ as $\boldsymbol{x}_{\boldsymbol{p}_j} = (\boldsymbol{x}^{cT}_{\boldsymbol{p}_j}, \boldsymbol{x}^{mT}_{\boldsymbol{p}_j})^T$, where $\boldsymbol{x}^c_{\boldsymbol{p}_j}$ consists of all the known variables that are associated with $p^i = 0$, $\boldsymbol{x}^m_{\boldsymbol{p}_j}$ consists of the variables with $p^i = 1$. For instance, if an incomplete data point $\boldsymbol{x}_{\boldsymbol{p}_j}$ follows a missing pattern $\boldsymbol{p}_j = (1,1,0,1,0)$, $\boldsymbol{x}^c_{\boldsymbol{p}_j} = (x^3, x^5)$, $\boldsymbol{x}^m_{\boldsymbol{p}_j} = (x^1, x^2, x^4)$.

The same logic applies to the hidden neurons' weights $\boldsymbol{w}$ in ELM. According to a missing pattern $\boldsymbol{p}_j$, $\boldsymbol{w}_{\boldsymbol{p}_j} = (\boldsymbol{w}^{cT}_{\boldsymbol{p}_j}, \boldsymbol{w}^{mT}_{\boldsymbol{p}_j})^T$, where $\boldsymbol{w}^c_{\boldsymbol{p}_j}$ are the weights for $\boldsymbol{x}^c_{\boldsymbol{p}_j}$, and $\boldsymbol{w}^m_{\boldsymbol{p}_j}$ for $\boldsymbol{x}^m_{\boldsymbol{p}_j}$. Therefore, the hidden neuron's output $h(\boldsymbol{x}_{\boldsymbol{p}_j})$ for $\boldsymbol{x}_{\boldsymbol{p}_j}$, is represented as

$$h(\boldsymbol{x}_{\boldsymbol{p}_j}) = h^c(\boldsymbol{x}^c_{\boldsymbol{p}_j}) + h^m(\boldsymbol{x}^m_{\boldsymbol{p}_j}) = \phi(\boldsymbol{x}^{cT}_{\boldsymbol{p}_j}\boldsymbol{w}^c_{\boldsymbol{p}_j}) + \phi(\boldsymbol{x}^{mT}_{\boldsymbol{p}_j}\boldsymbol{w}^m_{\boldsymbol{p}_j}), \tag{2}$$

For simplicity the bias term b is omitted. By applying equation 2, the matrix form of the hidden neurons' outputs $\mathbf{H}$ becomes $\mathbf{H} = \mathbf{H}^c + \mathbf{H}^m$, where

$$\mathbf{H}^c = \begin{pmatrix} h^c_1(\boldsymbol{x}^c_1) & \dots & h^c_L(\boldsymbol{x}^m_1) \\ \dots & \ddots & \dots \\ h^c_1(\boldsymbol{x}^c_N) & \dots & h^c_L(\boldsymbol{x}^m_N) \end{pmatrix}, \ \mathbf{H}^m = \begin{pmatrix} h^m_1(\boldsymbol{x}^c_1) & \dots & h^m_L(\boldsymbol{x}^m_1) \\ \dots & \ddots & \dots \\ h^m_1(\boldsymbol{x}^c_N) & \dots & h^m_L(\boldsymbol{x}^m_N) \end{pmatrix}. \tag{3}$$

In equation 3, the subscripts on h indicate the hidden neuron index, while the subscripts on $\boldsymbol{x}$ denote the sample index. In the above representations, $\boldsymbol{p}_j$, $\boldsymbol{x}^c_{\boldsymbol{p}_j}$, and both $\boldsymbol{w}^c_{\boldsymbol{p}_j}$ and $\boldsymbol{w}^m_{\boldsymbol{p}_j}$ are known values, while $\boldsymbol{x}^m_{\boldsymbol{p}_j}$ is unknown. Therefore, $h^m(\boldsymbol{x}^m)$ and $\mathbf{H}^m$ can not be computed directly, however, they can be approximated by other ELMs as discussed in the section 2.2.2.

2.2.2 The Secondary ELMs and The Primary ELM

Two types of ELMs are used in this model. The primary ELM is the single ELM that is predicting the original target values, $\hat{t} = \mathfrak{E}(\boldsymbol{x})$. In the primary ELM $\mathbf{H}^m$ is unknown. To approximate $\mathbf{H}^m$, secondary ELMs are built to learn an underlying functional relationship between the known values of $\boldsymbol{x}^c_{\boldsymbol{p}_j}$, $\boldsymbol{w}^m_{\boldsymbol{p}_j}$ and the unknown $h^m(\boldsymbol{x}^m_{\boldsymbol{p}_j})$, however, the missing pattern of each data point determines the predictability of $h^m(\boldsymbol{x}^m_{\boldsymbol{p}_j})$. Hence, a distinct functional relationship is considered for each missing pattern $\boldsymbol{p}_j$, $f_{\boldsymbol{p}_j} : (\boldsymbol{x}^c_{\boldsymbol{p}_j}, \boldsymbol{w}^m_{\boldsymbol{p}_j}) \to h^m(\boldsymbol{x}^m_{\boldsymbol{p}_j})$. A unique ELM is constructed for each $\boldsymbol{p}_j$ to learn such a function, $h^m(\boldsymbol{x}^m_{\boldsymbol{p}_j}) = \mathfrak{E}_{\boldsymbol{p}_j}(\boldsymbol{x}^c_{\boldsymbol{p}_j}, \boldsymbol{w}^m_{\boldsymbol{p}_j})$. These $\mathfrak{E}_{\boldsymbol{p}_j}$s are called the secondary ELMs. The total number of the possible missing patterns $\boldsymbol{p}_j$ is $2^d - 1$, if $\boldsymbol{x}$ has d variables. Hence in total, $2^d - 1$ secondary ELMs are constructed.

The secondary ELMs must be trained upon the complete data $\boldsymbol{X}_C$, and each secondary ELM must be trained upon different missing patterns, because of the distinct nature of each function $f_{\boldsymbol{p}_j}$. For each secondary ELM, $\mathfrak{E}_{\boldsymbol{p}_j}$, the complete data points $\boldsymbol{x} \in \boldsymbol{X}_C$ are split into $\boldsymbol{x}^c_{\boldsymbol{p}_j}$ and $\boldsymbol{x}^m_{\boldsymbol{p}_j}$ according to a missing pattern $\boldsymbol{p}_j$. Both $\boldsymbol{x}^c_{\boldsymbol{p}_j}$ and $\boldsymbol{x}^m_{\boldsymbol{p}_j}$ are known values in this case. $\boldsymbol{x}^m_{\boldsymbol{p}_j}$s are used to calculate the true target values $h^m(\boldsymbol{x}^m_{\boldsymbol{p}_j})$s as the training targets for the secondary ELMs. $\boldsymbol{x}^c_{\boldsymbol{p}_j}$, and $\boldsymbol{w}^m_{\boldsymbol{p}_j}$ are the input values in the secondary ELM training set.

$\boldsymbol{\mathfrak{X}}_i = [(\boldsymbol{x}^{cT}_1, \boldsymbol{w}^m_i), (\boldsymbol{x}^{cT}_2, \boldsymbol{w}^m_i), \ldots, (\boldsymbol{x}^{cT}_n, \boldsymbol{w}^m_i)]^T$ denotes the inputs for a secondary ELM $\mathfrak{E}_{\boldsymbol{p}_j}$. $\boldsymbol{\mathfrak{T}}^m_i = (h^m_i(\boldsymbol{x}^m_1), h^m_i(\boldsymbol{x}^m_2), ..., h^m_i(\boldsymbol{x}^m_n))^T$ denotes the outputs. i indicates the hidden neuron index in the primary ELM. Each trained secondary ELM is able to approximate the missing values of $h^m(\boldsymbol{x}^m_{\boldsymbol{p}_j})$. The approximation is $\hat{h}^m(\boldsymbol{x}^m_{\boldsymbol{p}_j}) = \mathfrak{E}_{\boldsymbol{p}_j}(\boldsymbol{x}^c_{\boldsymbol{p}_j}, \boldsymbol{w}^m_{\boldsymbol{p}_j})$, where in this step, the incomplete data $\boldsymbol{x}_{\boldsymbol{p}_j} \in \boldsymbol{X}_M$ is used. Finally, the approximated results for all $\boldsymbol{x} \in \boldsymbol{X}_M$ are generated by all the secondary ELMs, and are used to construct the complete $\mathbf{H}$ matrix in the primary ELM .

The complete $\mathbf{H}$ matrix in the primary ELM is obtained by adding the approximations from the secondary ELMs on top of the known values: $\mathbf{H}^* = \mathbf{H}^c + \hat{\mathbf{H}}^m$. Hence the $\boldsymbol{\beta}$ in the primary ELM can be solved by plugging in the $\mathbf{H}^*$ as $\boldsymbol{\beta}^* = (\mathbf{H}^{*T}\mathbf{H}^*)^{-1}\mathbf{H}^{*T}\mathbf{T}$. Therefore, the primary ELM can generate approximations for the original target $\mathbf{T}$.

3 Experiments

Experiments on two incomplete datasets have been conducted to examine the performance of the proposed method. The proposed method is compared with other imputation methods, including the mean imputation and the KNN imputation. The mean squared errors are adopted as the final measurements of the performances. To create fair comparison, one single primary ELM is built and is used to compute the MSEs for all three methods. For the mean imputation and the KNN imputation methods, the missing values are imputed first then tested using the primary ELM. For the Multi-ELM model, the missing values are not imputed and directly used to generate the final results from the primary ELM.

The dataset used in this experiments is summarized in the following table 1.

Table 1: Experiment Dataset

Name of the dataset	Sample size	Input Variables	Target Variable
Abalone [15]	2784	7	Age of Abalone
Wine Quality [16]	4898	11	Quality of The Wine

Both dataset have been divided into training set and validation set. Missing entrees have been created in the dataset randomly according to the percentage of missing values. 15% of missing values have been used for both Abalone and Wine Quality to compare the performance of algorithms across the different datasets. 5% of missing values is used for Abalone dataset to allow the comparison crossing different missing percentages. (see in table 2).

Table 2: Incomplete Dataset

Level of Incompleteness	5%		15%	
Name of the dataset	Abalone	Wine	Abalone	Wine
Incomplete Samples	1818	/	2318	4105
Complete Samples	966	/	466	793
Total Missing Entrees	974	/	2923	8081

Both the validation and the training sets are normalized to zero mean and unit standard deviation using the training set's mean and STD.

In the experiments, the configurations of the Primary and the Secondary ELMs are listed in the table 3.

Table 3: ELM Configurations

Level of Incompleteness	5%		15%	
Name of the dataset	Abalone	Wine	Abalone	Wine
Primary ELM Neurons	36	/	27	33
Secondary ELM Neurons	443	/	272	214

The experiments were conducted on the Sabine High performance computing cluster at the University of Houston, with 1 node 8 cores and 128G RAM.

The validation MSEs of all three methods are listed in the table 4.

Table 4: MSE Comparison

Level of Incompleteness	5%		15%	
Name of the dataset	Abalone	Wine	Abalone	Wine
Muilt-ELM Model	0.06	/	0.08	0.74
Mean Imputation	0.57	/	0.35	0.85
KNN Imputation	0.07	/	0.09	0.77

4 Conclusion

According to the performed experiments on the two datasets, for a reasonable amount of missing data the Multi-ELM model is outperforming the compared imputation methods. Even when the percentage of missing values increases to 15%, the Multi-ELM model has robust performance. In different data-based concepts, the Multi-ELM model decreases the noises from imputing the missing values directly. Furthermore, it can be concluded that the Multi-ELM model is capable of handling different missing patterns, even if they have never shown up in the training set. Multi-ELM model has the suitability for big data as it can be paralleled easily. In the future, the Multi-ELM model for incomplete data will be extended and tested for it's speed and performance on diverse datasets especially large datasets.

References

[1] Roderick J. A. Little and Donald B. Rubin. *Bayes and Multiple Imputation*, chapter 10, pages 200–220. John Wiley & Sons, Ltd, 2002.

[2] Alireza Farhangfar, Lukasz Kurgan, and Jennifer Dy. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41:3692–3705, 12 2008.

[3] Jerzy W Grzymala-Busse and Witold J Grzymala-Busse. Handling missing attribute values. In *Data mining and knowledge discovery handbook*, pages 33–51. Springer, 2009.

[4] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[5] Renjie Hu, Karl Ratner, Edward Ratner, Yoan Miche, Kaj-Mikael Björk, and Amaury Lendasse. Elm-som+: A continuous mapping for visualization. *Neurocomputing*, 365:147–156, 2019.

[6] Mortaza Jamshidian and Peter M Bentler. Ml estimation of mean and covariance structures with missing data using complete data routines. *Journal of Educational and behavioral Statistics*, 24(1):21–24, 1999.

[7] Renjie Hu, Amany Farag, Kaj-Mikael Björk, and Amaury Lendasse. Using machine learning to identify top predictors for nurses' willingness to report medication errors. *Array*, 8:100049, 2020.

[8] Eduardo R Hruschka, Estevam R Hruschka, and Nelson FF Ebecken. Evaluating a nearest-neighbor method to substitute continuous missing values. In *Australasian Joint Conference on Artificial Intelligence*, pages 723–734. Springer, 2003.

[9] Anton Akusok, Kaj-Mikael Björk, Yoan Miche, and Amaury Lendasse. High-performance extreme learning machines: A complete toolbox for big data applications. *IEEE Access*, 3:1011–1025, 2015.

[10] Renjie Hu, Edward Ratner, David Stewart, Kaj-Mikael Björk, and Amaury Lendasse. A modified lanczos algorithm for fast regularization of extreme learning machines. *Neurocomputing*, 414:172–181, 2020.

[11] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489 – 501, 2006. Neural Networks.

[12] Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Amaury Lendasse. Op-elm: Optimally pruned extreme learning machine. *IEEE Transactions on Neural Networks*, 21(1):158–162, 2010.

[13] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE, 2004.

[14] Guang-Bin Huang. What are extreme learning machines? filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle. *Cognitive Computation*, 7(3):263–278, Jun 2015.

[15] Arthur Asuncion and David Newman. Abalone. `https://archive.ics.uci.edu/ml/datasets/abalone`.

[16] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553, 2009.