Variable-Specific Recurrent Imputation for Multivariate Time Series with Missing Values

Qiuling Suo*, Liuyi Yao *, Guangxu Xun †, Jianhui Sun †, Aidong Zhang†
*Department of Computer Science and Engineering, State University of New York at Buffalo, NY, US
Email: {qiulings, liuyiyao}@buffalo.edu
†Department of Computer Science, University of Virginia, VA, US
Email: {gx5bt, js9gu, aidong}@virginia.edu

I. Introduction

Multivariate time series data are ubiquitous in real-world healthcare systems. It is a common issue that the data contain missing values due to various reasons, such as sensor damage, data corruption, patient dropout. There have been various works on filling the missing values in multivariate time series. Classical imputation methods include KNN-based, Matrix Factorization based, and Expectation-Maximization (EM) based imputation and so on. These methods are developed for general imputation purpose and rarely utilize the temporal relations between observations. Classical statistical time series models such as autoregressive (AR) models and dynamic linear models (DLM) (e.g. [1]) can capture the temporal information, but they are essentially linear and may not be suitable for modern complex large-scale data. ImputeTS [2] employs time dependencies on univariate time series imputation, which ignores feature correlations. Recent works [3, 4] develop the imputation framework that can take advantages of the traditional methods and resolve their drawbacks. Another trend of models is based on recurrent neural network (RNN) [5–10], utilizing RNN to capture temporal dependencies and further considering various aspects of the data characteristics, such as time decay, feature correlation, residual link, and temporal belief gate. In this paper, we propose an RNN-based imputation method for filling the missing values in multivariate time series. RNN is used to capture the temporal information of time series. We use a global RNN and variable-specific RNNs to perform imputation based on historical information, and a fusion gate to combine them. At each timestamp, we use a regression layer to impute the value of a certain variable using other variables, by utilizing the relationship of variables. Bi-directional imputation is adopted to improve the ability of long-term memory and performance of starting timestamps.

II. Methodology

A. Definitions

Suppose a multivariate time series (MTS) $\mathbf{X}$ is a sequence of T observations, and has D variables at each timestamp t, it can be denoted as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]^\top \in \mathbb{R}^{T\times D}$, where $\mathbf{x}_t = [x_t^1, x_t^2, ..., x_t^D]^\top \in \mathbb{R}^D$. In reality, $\mathbf{X}$ may carry missing values due to unexpected accidents. We introduce a masking vector $\mathbf{m}_t \in \{0,1\}^D$ to denote which variables are missing at time stamp t: if x_t^d is missing, $m_t^d = 0$; otherwise $m_t^d = 1$. In the healthcare datasets, the MTS are usually the measurement of vital signs and laboratory values. The overall framework is shown in Figure 1. In the model, the input vector $\mathbf{x}_{t-1}$ at time $t-1$ is feed into an RNN cell RNN_g and each variable in $\mathbf{x}_{t-1}$ also goes through an RNN separately to obtain latent representations $\mathbf{h}_t$ and $\{\mathbf{h}_t^d\}_{d=1}^D$, which contain the historical information. The latent representations are used to perform historical imputation. After that, each variable at time $t-1$ are predicted using other variables at the same timestamp, i.e. feature-level imputation. The imputed vector $\widehat{\mathbf{x}}_t$ is then used as the input of next RNN cells.

B. Global Imputation

RNN is used to memorize the historical characteristics of the input time series, and feature regression is used to learn the relationship among features. We feed the input vector at timestamp t to obtain a global representation which contains the information of all the variables as follows:

$$\mathbf{h}_t = \mathrm{RNN_g}(\mathbf{h}_{t-1}, \mathbf{x}_t), \tag{1}$$

where $\mathbf{h}_t \in \mathbb{R}^p$ is the hidden state that memorizes the historical information, p is the predefined size of hidden state, and $\mathbf{x}_t \in \mathbb{R}^D$ is the input vector of variables at time t. RNN() can be any RNN variant, such as LSTM [11] and GRU [12]. We use LSTM in our experiments. After obtaining the historical latent representation, we use a fully connected layer to predict variable values:

$$\widetilde{\mathbf{x}}_{gt} = \mathbf{W}_g^\top \mathbf{h}_{t-1} + \mathbf{b}_g, \tag{2}$$

where $\mathbf{W}_g \in \mathbb{R}^{p\times D}$ and $\mathbf{b}_g \in \mathbb{R}^D$ are model parameters to be learned. In this way, we can obtain the estimation $\widetilde{\mathbf{x}}_{gt}$ using historical observations.

C. Variable-Specific Imputation

In MTS, variables often have different patterns in terms of time, e.g. some variables remain stable during a long time interval on account of the homeostatic properties of human body; some variables may change dramatically due to the status of diseases. Using a global RNN model may not sufficiently capture the variable-specific information. Therefore, in addition to the global RNN, we use variable-specific

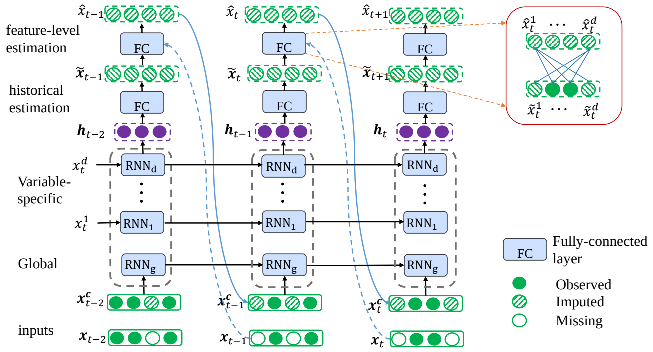

Fig. 1. Recurrent imputation network.

$\{\mathrm{RNN}_d\}_{d=1}^D$ to model time series for each variable separately. For a variable d, its latent representation is written as:

$$\mathbf{h}_t^d = \mathrm{RNN}_d(\mathbf{h}_{t-1}^d, x_t^d), \tag{3}$$

where $\mathbf{h}_t^d \in \mathbb{R}^q$ is the variable-specific representation for the time series of d-th variable, and q is the predefined size of hidden states. Similar to Eq. (2), we can obtain the estimated value for each variable:

$$\widetilde{x}_{st} = \mathbf{W}_s^\top \mathbf{h}_{t-1}^d + b_s, \tag{4}$$

where $\mathbf{W}_s^\top \in \mathbf{R}^{q\times 1}$ and $b_s \in \mathbb{R}$ are the parameters to be learned. The imputed values for all the parameters are $\widetilde{\mathbf{x}}_{st} = (\widetilde{x}_{1t}, \widetilde{x}_{2t}, ..., \widetilde{x}_{Dt}) \in \mathbb{R}^D$.

Given the global features $\mathbf{h}_t$ and variable-specific features $\mathbf{h}_t^d$ obtained by Eq. (1) and Eq. (3), we calculate a fusion rate $r_t^d \in \mathbb{R}$ for each variable at timestamp t:

$$r_t^d = \sigma(\mathbf{W}_{rg}^\top \mathbf{h}_t + \mathbf{W}_{rd}^\top \mathbf{h}_t^d + b), \tag{5}$$

where $\mathbf{W}_{rg} \in \mathbb{R}^p$, $\mathbf{W}_{rd} \in \mathbb{R}^q$ and $b \in \mathbb{R}$ are the parameters to be learned, and $\sigma()$ is the activation function to rescale the value to the range of [0,1]. The fusion rate is used to merge the estimations from global RNN and specific RNN, as follows:

$$\widetilde{\mathbf{x}}_t = \mathbf{r}_t \odot \widetilde{\mathbf{x}}_{gt} + (1 - \mathbf{r}_t) \odot \widetilde{\mathbf{x}}_{st}, \tag{6}$$

where $\mathbf{r}_t = [r_t^1, r_t^2, ..., r_t^D] \in \mathbb{R}^D$, and $\odot$ is the element-wise multiplication operator. In this way, we can obtain the historical imputation $\widetilde{x}$ that can fully utilize the multivariate information and variable-specific characteristics.

D. Feature-level Imputation

The above parts learn the representation utilizing the historical information. After the above process, we can obtain a complete vector $\widetilde{\mathbf{x}}_t^c = \mathbf{m}_t \mathbf{x}_t + (1 - \mathbf{m}_t)\widetilde{\mathbf{x}}_t$, whose missing values are filled by $\widetilde{\mathbf{x}}_t$. However, features have correlations with each other, which can be useful in the imputation process. Following [10], we add a regression layer to predict a certain missing value based on the other features at time t and obtain a feature-based imputation vector $\widetilde{\mathbf{x}}_{ft}$ as follows,

$$\widetilde{\mathbf{x}}_{ft} = \mathbf{W}_f \widetilde{\mathbf{x}}_t^c + \mathbf{b}_f, \tag{7}$$

where $\mathbf{W}_f \in \mathbf{W}^{D\times D}$ and $\mathbf{b}_f \in \mathbf{R}^D$ are model parameters to be learned. We then combine the historical and feature-based estimation to obtain $\widehat{x}_t$.

The imputation loss is the mean square error between the estimated values and observed ones, denoted as $\mathcal{L}_{mse}(\mathbf{x}_t, \widehat{\mathbf{x}}_t) = \sum \|\mathbf{m}_t(\mathbf{x}_t - \widehat{\mathbf{x}}_t)\|^2$. To accelerate the convergence speed, we accumulate all the estimation errors of the estimations $\widehat{\mathbf{x}}_t, \widetilde{\mathbf{x}}_t, \widetilde{\mathbf{x}}_{ft}$:

$$\mathcal{L}_t = \mathcal{L}_{mse}(\mathbf{x}_t, \widehat{\mathbf{x}}_t) + \mathcal{L}_{mse}(\mathbf{x}_t, \widetilde{\mathbf{x}}_t) + \mathcal{L}_{mse}(\mathbf{x}_t, \widetilde{\mathbf{x}}_{ft}) \tag{8}$$

E. Bi-directional RNN

Although RNN provides an elegant way to model sequential data, it may not be sufficiently trained when the sequence is long and time interval between two observations is large. Especially, it may fail when the first few time stamps need to be imputed, as few information is available in the starting part. To enable future information to be accessible, we employ bidirectional-RNN (BRNN) to estimate the variables from both forward and backward directions. Following Section II-B to II-D, we can obtain the imputed values $\widehat{\mathbf{x}}_t$ and $\widehat{\mathbf{x}}_t'$ from forward and backward imputation respectively. The final estimation in the t-th timestamp is the mean of $\widehat{\mathbf{x}}_t$ and $\widehat{\mathbf{x}}_t'$.

III. Experiments

A. Implementation Details

We first normalize the values before feeding them into the developed framework. The imputation framework is implemented with Pytorch. Patients have different numbers of recorded time points, and the length of MTS varies from 10 to 260. To enable mini-batch optimization, we pad zeros to make the training samples in the same batch have a fixed sequence length, which is the maximum length in a certain mini-batch. We use Adam optimizer to optimize the model parameters.

B. Results

TABLE I
Imputation performance in terms of nRMSD.

	PCL	PK	PLCO2	PNA	HCT	HGB	MCV
Forward	0.2609	0.3094	0.2725	0.2826	0.2957	0.3009	0.3204
Mean	0.2951	0.2767	0.3009	0.2932	0.2873	0.2924	0.3095
KNN	0.2204	0.2491	0.2415	0.2283	0.2200	0.2205	0.2669
3DMICE	0.1996	0.2613	0.2326	0.2136	0.1447	0.1429	0.2679
Proposed_1	0.1715	0.2427	0.2081	0.1878	0.1488	0.1496	0.4016
Proposed_2	**0.1550**	**0.2298**	**0.1960**	**0.1736**	**0.0901**	**0.0875**	**0.2455**
	PLT	WBC	RDW	PBUN	PCRE	PGLU	Avg.
Forward	0.2569	0.2904	0.2912	0.2416	0.2797	0.3316	0.2873
Mean	0.3185	0.2995	0.3175	0.3127	0.3065	0.2818	0.2994
KNN	0.2477	0.2514	0.2527	0.2379	0.2436	0.2639	0.2418
3DMICE	0.2259	0.2554	0.2492	0.1848	.2291	0.2768	0.2218
Proposed_1	0.2826	0.2607	0.3729	0.2390	0.2957	0.2611	0.2479
Proposed_2	**0.1749**	**0.2115**	**0.2076**	**0.1539**	**0.2099**	**0.2523**	**0.1837**

The performance is evaluated using the dataset provided in ICHI 2019 data analysis challenge [1]. We report the results of the imputation accuracy using the provided validation mask. The evaluation metric is nRMSD as defined in [4]. The imputation performance for the 13 variables is shown in Table I. The baseline methods include: mean, forward imputation, knn and 3D-MICE. In forward imputation, the missing values are filled with the last observed value of the same variable, and if no value is observed before the missing one, the mean is used to impute. In Proposed_1, the normalization of inputs uses global mean/std of entire patients, and in proposed_2, we use

[1] http://www.ieee-ichi.org/2019/challenge.html

the mean/std of each patient each variable. The two methods are both based on our proposed method, but only differ in the way of normalizing the input data. We should point out that since the imputed data is evaluated in the normalized space, the way of normalizing the training data does matter a lot. Since the evaluation metric nRMSD uses the normalization factor per patient per variable, proposed_2 is much better than proposed_1.

References

[1] C. F. Ansley and R. Kohn, "On the estimation of arima models with missing values," pp. 9–37, 1984.

[2] S. Moritz and T. Bartz-Beielstein, "imputets: time series missing value imputation in r," *The R Journal*, 2017.

[3] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Advances in neural information processing systems*, 2016, pp. 847–855.

[4] Y. Luo, P. Szolovits, A. S. Dighe, and J. M. Baron, "3d-mice: integration of cross-sectional and longitudinal imputation for multi-analyte longitudinal clinical data," *Journal of the American Medical Informatics Association*, 2017.

[5] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, p. 6085, 2018.

[6] J. Yoon, W. R. Zame, and M. van der Schaar, "Multi-directional recurrent neural networks: A novel method for estimating missing data," 2017.

[7] L. Shen, Q. Ma, and S. Li, "End-to-end time series imputation via residual short paths," in *Asian Conference on Machine Learning*, 2018, pp. 248–263.

[8] Y.-J. Kim and M. Chi, "Temporal belief memory: Imputing missing data during rnn training." in *In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-2018)*, 2018.

[9] Y. Luo, X. Cai, Y. Zhang, J. Xu *et al.*, "Multivariate time series imputation with generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 1596–1607.

[10] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "Brits: Bidirectional recurrent imputation for time series," in *Advances in Neural Information Processing Systems*, 2018, pp. 6775–6785.

[11] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.