BRITS

Armine Hayrapetyan

Bidirectional **R**ecurrent **I**mputation for **T**ime **S**eries

Given multiple (possibly correlated) time series data, it is important to fill in missing values and at the same time to predict their class labels. Existing imputation methods often impose strong assumptions of the underlying data generating process, such as linear dynamics in the state space. In this paper, we propose a novel method, called BRITS, based on recurrent neural networks for missing value imputation in time series data.

Bidirectional **R**ecurrent **I**mputation for **T**ime **S**eries

Our proposed method directly learns the missing values in a bidirectional recurrent dynamical system, without any specific assumption. The imputed values are treated as variables of RNN graph and can be effectively updated during backpropagation.

Advantages of BRITS

1. it can handle multiple correlated missing values in time series
2. it generalizes to time series with nonlinear dynamics underlying
3. it provides a data-driven imputation procedure and applies to general settings with missing data

Introduction

Much prior work proposed to fix the missing data problem with statistics and machine learning approaches. However most of them require fairly strong assumptions on missing values. We can fill the missing values using classical statistical time series models such as ARMA or ARIMA.
But these models are essentially linear (after differencing).

In this paper, we propose BRITS, a novel method for filling the missing values for multiple correlated time series. Internally, BRITS adapts recurrent neural networks (RNN) for imputing missing values, without any specific assumption over the data.

Introduction

In particular, we make the following technical contributions:

- We design a bidirectional RNN model for imputing missing values. We directly use RNN for predicting missing values, instead of tuning weights for smoothing as in [10]. Our method does not impose specific assumption, hence works more generally than previous methods
- We regard missing values as variables of the bidirectional RNN graph, which are involved in the backpropagation process. In such case, missing values get delayed gradients in both forward and backward directions with consistency constraints, which makes the estimation of missing values more accurate

Contributions

- We simultaneously perform missing value imputation and classification/regression of applications jointly in one neural graph. This alleviates the error propagation problem from imputation to classification/regression and makes the classification/regression more accurate.
- We evaluate our model on three real-world datasets, including an air quality dataset, a health-care dataset and a localization dataset of human activities. Experimental results show that our model outperforms the state-of-the-art models for both imputation and classification/regression accuracies.

Preliminary

Definition 1 (Multivariate Time Series) *We denote a multivariate time series* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ *as a sequence of* T *observations. The* t*-th observation* $\mathbf{x}_t \in \mathbb{R}^D$ *consists of* D *features* $\{x_t^1, x_t^2, \ldots, x_t^D\}$*, and was observed at timestamp* s_t *(the time gap between different timestamps may not be the same). In reality, due to unexpected accidents, such as equipment damage or communication error,* $\mathbf{x}_t$ *may have the missing values (e.g., in Fig. 1,* x_1^3 *in* $\mathbf{x}_1$ *is missing). To represent the missing values in* $\mathbf{x}_t$*, we introduce a* masking vector $\mathbf{m}_t$ *where,*

$$\mathbf{m}_t^d = \begin{cases} 0 & \text{if } x_t^d \text{ is not observed} \\ 1 & \text{otherwise} \end{cases} .$$

In many cases, some features can be missing for consecutive timestamps (e.g., blue blocks in Fig. 1). We define δ_t^d *to be the time gap from the last observation to the current timestamp* s_t*, i.e.,*

$$\delta_t^d = \begin{cases} s_t - s_{t-1} + \delta_{t-1}^d & \text{if } t > 1, \mathbf{m}_{t-1}^d = 0 \\ s_t - s_{t-1} & \text{if } t > 1, \mathbf{m}_{t-1}^d = 1 \\ 0 & \text{if } t = 1 \end{cases} .$$

Preliminary

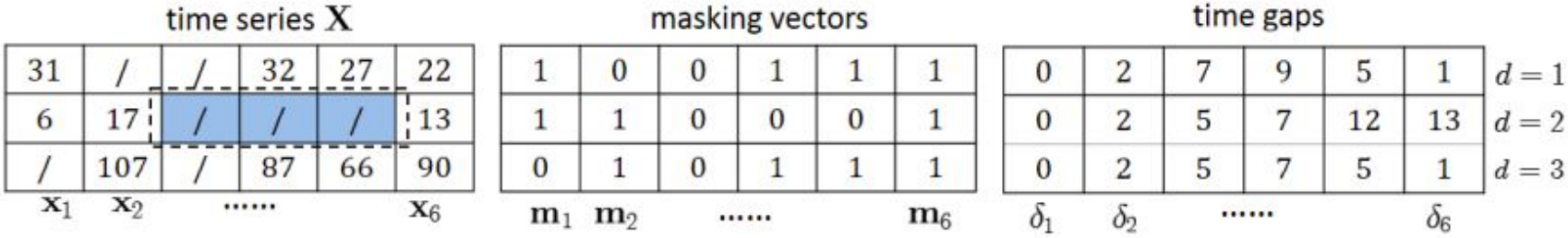

Figure 1: An example of multivariate time series with missing values. $\mathbf{x}_1$ to $\mathbf{x}_6$ are observed at $s_{1...6} = 0, 2, 7, 9, 14, 15$ respectively. Considering the 2nd feature in $\mathbf{x}_6$, the last observation of the 2nd feature took place at $s_2 = 2$, and we have that $\delta_6^2 = s_6 - s_2 = 13$.

BRITS

We start the description with the simplest case: the variables observed at the same time are mutually uncorrelated. For such case, we propose algorithms for imputation with unidirectional recurrent dynamics and bidirectional recurrent dynamics, respectively.

Unidirectional Uncorrelated Recurrent Imputation

For the simplest case, we assume that for the t-th step, x_t^i and x_t^j are uncorrelated if $i \neq j$ (but x_t^i may be correlated with some $x_{t' \neq t}^j$). We first propose an imputation algorithm by unidirectional recurrent dynamics, denoted as **RITS-I**.

For the t-th step, if $\mathbf{x}_t$ is actually observed, we use it to validate our imputation and pass $\mathbf{x}_t$ to the next recurrent steps. Otherwise, since the future observations are correlated with the current value, we replace $\mathbf{x}_t$ with the obtained imputation, and validate it by the future observations.

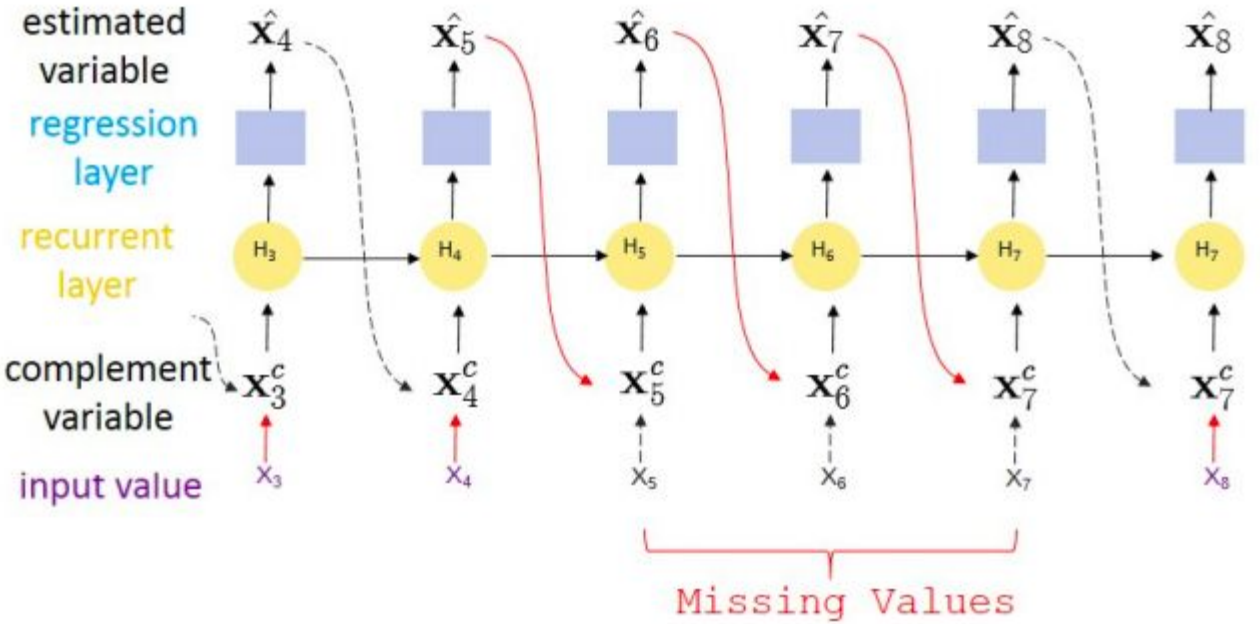

Figure 2: Imputation with unidirectional dynamics.

Example 1 *Suppose we are given a time series* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{10}\}$, *where* $\mathbf{x}_5, \mathbf{x}_6$ *and* $\mathbf{x}_7$ *are missing*[2]. *According to the recurrent dynamics, at each time step* t, *we can obtain an estimation* $\hat{\mathbf{x}}_t$ *based on the previous* $t-1$ *steps. In the first* 4 *steps, the estimation error can be obtained immediately by calculating the estimation loss function* $\mathcal{L}_e(\hat{\mathbf{x}}_t, \mathbf{x}_t)$ *for* $t = 1, \ldots, 4$. *However, when* $t = 5, 6, 7$, *we cannot get the error immediately since the true values are missing. Nevertheless, note that at the* 8*-th step,* $\hat{\mathbf{x}}_8$ *depends on* $\hat{\mathbf{x}}_5$ *to* $\hat{\mathbf{x}}_7$. *We thus obtain a "delayed" error for* $\hat{\mathbf{x}}_{t=5,6,7}$ *at the* 8*-th step.*

Algorithm

We introduce a recurrent component and a regression component for imputation. The recurrent component is achieved by a recurrent neural network and the regression component is achieved by a fully-connected network. A standard recurrent network [17] can be represented as $\mathbf{h_t = sigma(W_h\, h_{t-1} + U_h\, x_t + b_h)}$

where σ is the sigmoid function, $\mathbf{W_h}$, $\mathbf{U_h}$ and $\mathbf{b_h}$ are parameters, and $\mathbf{h_t}$ is the hidden state of previous time steps.

Algorithm

In our case, since $\mathbf{x}_t$ may have missing values, we cannot use $\mathbf{x}_t$ as the input directly as in the above equation. Instead, we use a “complement" input $\mathbf{x}^c_t$ derived by our algorithm when $\mathbf{x}_t$ is missing. Formally, we initialize the initial hidden state $\mathbf{h}_0$ as an all-zero vector and then update the model by:

$$\hat{\mathbf{x}}_t = \mathbf{W}_x \mathbf{h}_{t-1} + \mathbf{b}_x, \quad (1)$$

$$\mathbf{x}^c_t = \mathbf{m}_t \odot \mathbf{x}_t + (1 - \mathbf{m}_t) \odot \hat{\mathbf{x}}_t, \quad (2)$$

$$\gamma_t = \exp\{-\max(0, \mathbf{W}_\gamma \delta_t + \mathbf{b}_\gamma)\}, \quad (3)$$

$$\mathbf{h}_t = \sigma(\mathbf{W}_h[\mathbf{h}_{t-1} \odot \gamma_t] + \mathbf{U}_h[\mathbf{x}^c_t \circ \mathbf{m}_t] + \mathbf{b}_h), \quad (4)$$

$$\ell_t = \langle \mathbf{m}_t, \mathcal{L}_e(\mathbf{x}_t, \hat{\mathbf{x}}_t) \rangle . \quad (5)$$

Algorithm

Eq. (1) is the regression component which transfers the hidden state $\mathbf{h}_{t-1}$ to the estimated vector $\hat{\mathbf{x}}_t$. In Eq. (2), we replace missing values in $\mathbf{x}_t$ with corresponding values in $\hat{\mathbf{x}}_t$, and obtain the complement vector xct. Besides, since the time series may be irregularly sampled, in Eq. (3), we further introduce a temporal decay factor $\boldsymbol{\gamma}_t$. Such factor represents the missing patterns in the time series which is critical to imputation [10]. In Eq. (4), based on the decayed hidden state, we predict the next state $\mathbf{h}_t$. Here, ◦ indicates the concatenate operation. In the mean time, we calculate the estimation error by the estimation loss function $\mathbf{L}_e$ in Eq. (5). In our experiment, we use the mean absolute error for $\mathbf{L}_e$. Finally, we predict the task label **y** as

$$\hat{\mathbf{y}} = f_{out}(\sum_{i=1}^{T} \alpha_i \mathbf{h}_i),$$

Bidirectional **U**ncorrelated **R**ecurrent **I**mputation

In the RITS-I, errors of estimated missing values are delayed until the presence of the next observation. For example, in Example 1, the error of $\hat{\mathbf{x}}_5$ is delayed until $\mathbf{x}_8$ is observed. Such error delay makes the model converge slowly and in turn leads to inefficiency in training. In the meantime, it also leads to the bias exploding problem [6], i.e., the mistakes made early in the sequential prediction are fed as input to the model and may be quickly amplified. In this section, we propose an improved version called BRITS-I. The algorithm alleviates the above-mentioned issues by utilizing the bidirectional recurrent dynamics on the given time series, i.e., besides the forward direction, each value in time series can be also derived from the backward direction by another fixed arbitrary function.

To illustrate the intuition of BRITS-I algorithm, again, we consider Example 1. Consider the backward direction of the time series. In bidirectional recurrent dynamics, the estimation $\hat{\mathbf{x}_4}$ reversely depends on $\hat{\mathbf{x}_5}$ to $\hat{\mathbf{x}_7}$. Thus, the error in the 5-th step is back-propagated from not only the 8-th step in the forward direction (which is far from the current position), but also the 4-th step in the backward direction (which is closer). Formally, the BRITS-I algorithm performs the RITS-I as shown in Eq. (1) to Eq. (5) in forward and backward directions, respectively. In the forward direction, we obtain the estimation sequence $\{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_T\}$ and the loss sequence $\{\ell_1, \ell_2, \ldots, \ell_T\}$. Similarly, in the backward direction, we obtain another estimation sequence $\{\hat{\mathbf{x}}'_1, \hat{\mathbf{x}}'_2, \ldots, \hat{\mathbf{x}}'_T\}$ and another loss sequence $\{\ell_1{}', \ell_2{}', \ldots, \ell_T{}'\}$. We enforce the prediction in each step to be consistent in both directions by introducing the "*consistency loss*":

$$\ell_t^{cons} = \text{Discrepancy}(\hat{\mathbf{x}}_t, \hat{\mathbf{x}}'_t) \tag{6}$$

where we also use the mean absolute error as the discrepancy in our experiment. The final estimation loss is obtained by accumulating the forward loss ℓ_t, the backward loss ℓ'_t and the consistency loss ℓ_t^{cons}. The final estimation in the t-th step is the mean of $\hat{\mathbf{x}_t}$ and $\hat{\mathbf{x}_t}{}'$.

Correlated Recurrent Imputation

The previously proposed algorithms RITS-I and BRITS-I assume that features observed at the same time are mutually uncorrelated. This may be not true in some scenarios. For example, in the air quality data [32], each feature represents one measurement in a monitoring station. Obviously, the observed measurements are spatially correlated. In general, the measurement in one monitoring station is close to those values observed in its neighboring stations. In this case, we can estimate a missing measurement according to both its historical data, and the measurements in its neighbors.

BRITS

We refer to $\mathbf{x}^{\wedge}_t$ as a “history-based estimation". In this section, we derive another “feature-based estimation" for each $\mathbf{x}^{\mathbf{d}}_t$, based on the other features at time $\mathbf{s}_t$. Specifically, at the t-th step, we first obtain the complement observation $\mathbf{x}^{\mathbf{c}}_t$ by Eq. (1) and Eq. (2). Then, we define our feature-based estimation as $\mathbf{z}^{\wedge}_t$ where

$\mathbf{z}^{\wedge}_t = \mathbf{W}_z\mathbf{x}^{\mathbf{c}}_t + \mathbf{b}_z$. We further combine the historical-based estimation $\mathbf{x}^{\wedge}_t$ and the feature-based estimation $\mathbf{z}^{\wedge}_t$. We denote the combined vector as $\mathbf{c}^{\wedge}_t$, and we have that

$$\beta_t = \sigma(\mathbf{W}_\beta[\gamma_t \circ \mathbf{m}_t] + \mathbf{b}_\beta) \quad (8)$$

$$\hat{\mathbf{c}}_t = \beta_\mathbf{t} \odot \hat{\mathbf{z}}_t + (1 - \beta_\mathbf{t}) \odot \hat{\mathbf{x}}_t. \quad (9)$$

BRITS

Here we use $\beta_t \in [0,1]^D$ as the weight of combining the history-based estimation $\hat{\mathbf{x}}_t$ and the feature-based estimation $\hat{\mathbf{z}}_t$. Note that $\hat{\mathbf{z}}_t$ is derived from $\mathbf{x}_t^c$ by Eq. (7). The elements of $\mathbf{x}_t^c$ can be history-based estimations or truly observed values, depending on the presence of the observations. Thus, we learn the weight β_t by considering both the temporal decay γ_t and the masking vector $\mathbf{m}_t$ as shown in Eq. (8). The rest parts are similar as the feature uncorrelated case. We first replace missing values in $\mathbf{x}_t$ with the corresponding values in $\hat{\mathbf{c}}_t$. The obtained vector is then fed to the next recurrent step to predict memory $\mathbf{h}_t$:

$$\mathbf{c}_t^c = \mathbf{m}_t \odot \mathbf{x}_t + (1 - \mathbf{m}_t) \odot \hat{\mathbf{c}}_t \tag{10}$$

$$\mathbf{h}_t = \sigma(\mathbf{W}_h[\mathbf{h}_{t-1} \odot \gamma_t] + \mathbf{U}_h[\mathbf{c}_t^c \circ \mathbf{m}_t] + \mathbf{b}_h). \tag{11}$$

However, the estimation loss is slightly different with the feature uncorrelated case. We find that simply using $\ell_t = \mathcal{L}_e(\mathbf{x}_t, \hat{\mathbf{c}}_t)$ leads to a very slow convergence speed. Instead, we accumulate all the estimation errors of $\hat{\mathbf{x}}_t$, $\hat{\mathbf{z}}_t$ and $\hat{\mathbf{c}}_t$:

$$\ell_t = \mathcal{L}_e(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \mathcal{L}_e(\mathbf{x}_t, \hat{\mathbf{z}}_t) + \mathcal{L}_e(\mathbf{x}_t, \hat{\mathbf{c}}_t).$$

Experiment / Air Quality Data

We evaluate our models on the air quality dataset, which consists of PM2.5 measurements from 36 monitoring stations in Beijing. The measurements are hourly collected from 2014/05/01 to 2015/04/30. Overall, there are 13.3% values are missing. For this dataset, we do pure imputation task. We use exactly the same train/test setting as in prior work [32], i.e., we use the 3rd, 6th, 9th and 12th months as the test data and the other months as the training data. To train our model, we randomly select every 36 consecutive steps as one time series.

Experiment/ Health-Care Data

We evaluate our models on health-care data in PhysioNet Challenge 2012 [27], which consists of 4000 multivariate clinical time series from intensive care unit (ICU). Each time series contains 35 measurements such as Albumin, heart-rate etc., which are irregularly sampled at the first 48 hours after the patient's admission to ICU. We stress that this dataset is extremely sparse. There are up to 78% missing values in total. For this dataset, we do both the imputation task and the classification task. To evaluate the imputation performance, we randomly eliminate 10% of observed measurements from data and use them as the ground-truth. At the same time, we predict the in-hospital death of each patient as a binary classification task. Note that the eliminated measurements are only used for validating the imputation, and they are never visible to the model.

Experiment / Localization for Human Activity Data

The UCI localization data for human activity [18] contains records of five people performing different activities such as walking, falling, sitting down etc (there are 11 activities). Each person wore four sensors on her/his left/right ankle, chest, and belt. Each sensor recorded a 3-dimensional coordinates for about 20 to 40 millisecond. We randomly select 40 consecutive steps as one time series, and there are 30, 917 time series in total. For this dataset, we do both imputation and classification tasks. Similarly, we randomly eliminate 10% observed data as the imputation ground-truth. We further predict the corresponding activity of observed time series (i.e., walking, sitting, etc).

Model Implementations

To make a fair comparison, we control the number of parameters of all models as around 80, 000. We train our model by an Adam optimizer with learning rate 0.001 and batch size 64. For all the tasks, we normalize the numerical values to have zero mean and unit variance for stable training.

We use different early stopping strategies for pure imputation task and classification tasks. For the imputation tasks, we randomly select 10% of non-missing values as the validation data. The early stopping is thus performed based on the validation error. For the classification tasks, we first pre-train the model as a pure imputation task and report its imputation accuracy. Then we use 5-fold cross validation to further optimize both the imputation and classification losses simultaneously.

Baseline Methods

- **Mean**: We simply replace the missing values with corresponding global mean.
- **KNN**: We use k-nearest neighbor [13] (with normalized Euclidean distance) to find the similar samples, and impute the missing values with weighted average of its neighbors.
- **Matrix Factorization (MF)**: We factorize the data matrix into two low-rank matrices, and fill the missing values by matrix completion [13].
- **MICE**: We use Multiple Imputation by Chained Equations (MICE), a widely used imputation method, to fill the missing values. MICE creates multiple imputations with chained equations [2].

Baseline Methods

- **ImputeTS**: We use ImputeTS package in R to impute the missing values. ImputeTS is a widely used package for missing value imputation, which utilizes the state space model and kalman smoothing [23].
- **STMVL**: Specifically, we use STMVL for the air quality data imputation. STMVL is the state-of-the-art method for air quality data imputation. It further utilizes the geo-locations when imputing missing values [32]

Table 1: Performance Comparison for Imputation Tasks (in MAE(MRE%))

Method		Air Quality	Health-care	Human Activity
Non-RNN	Mean	55.51 (77.97%)	0.461 (65.61%)	0.767 (96.43%)
	KNN	29.79 (41.85%)	0.367 (52.15%)	0.479 (58.54%)
	MF	27.94 (39.25%)	0.468 (67.97%)	0.879 (110.44%)
	MICE	27.42 (38.52%)	0.510 (72.5%)	0.477 (57.94%)
	ImputeTS	19.58 (27.51%)	0.390 (54.2%)	0.363 (45.65%)
	STMVL	12.12 (17.40%)	/	/
RNN	GRU-D	/	0.559 (77.58%)	0.558 (70.05%)
	M-RNN	14.05 (20.16%)	0.445 (61.87%)	0.248 (31.19%)
Ours	RITS-I	12.45 (17.93%)	0.385 (53.41%)	0.240 (30.10%)
	BRITS-I	11.58 (16.66%)	0.361 (50.01%)	0.220 (27.61%)
	RITS	12.19 (17.54%)	0.292 (40.82%)	0.248 (31.21%)
	BRITS	**11.56 (16.65%)**	**0.278 (38.72%)**	**0.219 (27.59%)**

Table 2: Performance Comparison for Classification Tasks

Method	Health-care (AUC)	Human Activity (Accuracy)
GRU-D	0.834 ± 0.002	0.940 ± 0.010
M-RNN	0.817 ± 0.003	0.938 ± 0.010
RITS-I	0.821 ± 0.007	0.934 ± 0.008
BRITS-I	0.831 ± 0.003	0.940 ± 0.012
RITS	0.840 ± 0.004	0.968 ± 0.010
BRITS	$\mathbf{0.850 \pm 0.002}$	$\mathbf{0.969 \pm 0.008}$

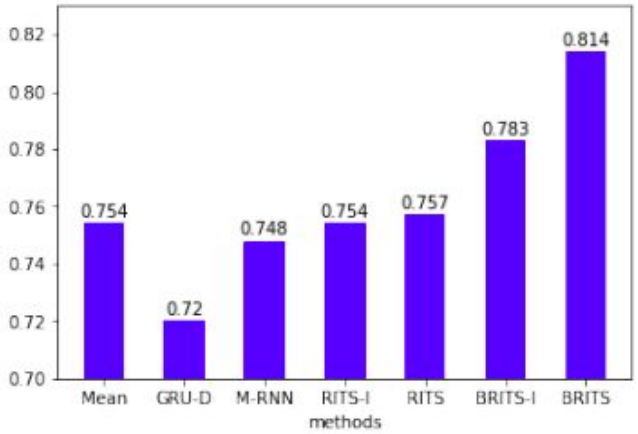

Figure 3: Health-care Classification Based on Different Imputations with Random Forest

Conclusion

In this paper, we proposed BRITS, a novel method to use recurrent dynamics to effectively impute the missing values in multivariate time series. Instead of imposing assumptions over the data generating process, our model directly learns the missing values in a bidirectional recurrent dynamical system, without any specific assumption. Our model treats missing values as variables of the bidirectional RNN graph. Thus, we get the delayed gradients for missing values in both forward and backward directions, which makes the imputation of missing values more accurate. We performed the missing value imputation and classification/regression simultaneously within a joint neural network. Experiment results show that our model demonstrates more accurate results for both imputation and classification/regression than state-of-the-art methods.