A Multi-Task Imputation and Classification Neural Architecture for Early Prediction of Sepsis from Multivariate Clinical Time Series

Yale Chang*, Jonathan Rubin*, Gregory Boverman, Shruti Vij, Asif Rahman, Annamalai Natarajan, Saman Parvaneh

Philips Research North America, Cambridge, USA
** Authors contributed equally*

Abstract

Early prediction of sepsis onset can notify clinicians to provide timely interventions to patients to improve their clinical outcomes. The key question motivating this work is: given a retrospective patient cohort consisting of multivariate clinical time series (e.g., vital signs and lab measurement) and patients' demographics, how to build a model to predict the onset of sepsis six hours earlier? To tackle this challenge, we first used a recurrent imputation for time series (RITS) approach to impute missing values in multivariate clinical time series. Second, we applied temporal convolutional networks (TCN) to the RITS-imputed data. Compared to other sequence prediction models, TCN can effectively control the size of sequence history. Third, when defining the loss function, we assigned custom time-dependent weights to different types of errors. We achieved 9th place (team name = prna, utility score = 0.328) at the 2019 PhysioNet Computing in Cardiology Challenge, which evaluated our proposed model on a real-world sepsis patient cohort. At a follow-up 'hackathon' event, held by the challenge organizers, an improved version of our algorithm achieved 2nd place (utility score = 0.342).

1. Introduction

The goal of the 2019 PhysioNet Computing in Cardiology Challenge was the early detection of sepsis using physiological data [1]. Sepsis is a life-threatening condition that puts 1.7 million lives at risk every year in the United States alone [2]. Early detection of sepsis in the intensive care unit would allow faster administration of antibiotic treatment and improve patient outcomes, as well as significantly reduce hospital expenses.

Automated systems that can accurately predict sepsis early based on available clinical data can allow faster treatment times and significant value for patients and care givers. However, there are various challenges associated with developing early prediction systems. First, predictions that are too early or false positive predictions can end up straining hospital resources, as well as being disadvantageous to patients. Second, predictions that are too late may reduce the impact of any administered intervention. Third, accuracy of predictions are affected by the nature of clinical time series data, which is messy and consists of irregularly sampled and missing data points.

Previous works on sepsis prediction have employed time varying Cox proportional hazards models [3], gradient boosting [4] and Gaussian process RNNs [5]. To tackle the challenges associated with this problem, we build a multi-task neural architecture for early sepsis detection. First, we define a loss function to approximate the provided custom utility function [1], penalizing both too early and too late detection of sepsis. Second, we apply a recurrent imputation for time series (RITS) model [6] to impute missing values in multivariate clinical time series. Compared to other commonly-used imputation techniques, RITS can achieve significantly lower imputation error. By combining the RITS reconstruction loss with the utility loss, the imputed feature values are adaptive to the prediction task. Third, we train a TCN model using the RITS-imputed data to maximize the approximated utility objective. Experimental results on the PhysioNet Challenge 2019 patient cohort demonstrate the effectiveness of our proposed approach.

2. Imputation Network

The input to our neural architecture is a matrix, X, that consists of multivariate clinical time series (including vital signs, laboratory values and demographic information) that can contain missing values ($-$). Each row, in X records the value of a feature and the columns $x_1, x_2, \dots x_t \in X$ record the value of each feature over time.

From X, we further derive two input matrices: 1) a matrix of masking vectors, m, that indicates whether the measurement is available (1) or missing (0); and 2) a $\boldsymbol{\delta}$ matrix that records the time (in hours) since the last available fea-

ISSN: 2325-887X DOI: 10.22489/CinC.2019.110

ture measurement. Examples of input matrix X, masking matrix m, and matrix $\boldsymbol{\delta}$ are shown in Equations (1) – (4).

As the input vector, x_t, may contain missing values, it cannot be fed directly into a recurrent neural network architecture. Instead, we first process x_t using a RITS network based on [6]. The RITS network uses a combination of *history-based estimation* and *feature-based estimation* to impute missing values. **History-based estimation** predicts missing feature values using the hidden state vector of a recurrent neural network. **Feature-based estimation** uses information about the available surrounding features at the current time, x_t, to make a prediction about missing variables.

$$X = \begin{array}{c} \\ \boldsymbol{x^1} \\ \vdots \\ \boldsymbol{x^d} \end{array} \begin{array}{c} \begin{array}{ccccc} \boldsymbol{x_1} & \boldsymbol{x_2} & \boldsymbol{x_3} & \cdots & \boldsymbol{x_t} \end{array} \\ \begin{bmatrix} 101 & 100 & - & 98 & 80 \\ 98.5 & - & - & 97 & 99 \\ 37 & - & - & - & - \end{bmatrix} \end{array} \tag{1}$$

$$\mathbf{m_t^d} = \begin{cases} 0 & \text{if } x_t^d \text{ is unobserved} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

$$\delta_{\mathbf{t}}^{\mathbf{d}} = \begin{cases} 1 + \delta_{\mathbf{t-1}}^{\mathbf{d}} & \text{if } t > 1, \mathbf{m_{t-1}^d} = \mathbf{0} \\ 1 & \text{if } t > 1, \mathbf{m_{t-1}^d} = \mathbf{1} \\ 0 & \text{if } t = 1 \end{cases} \tag{3}$$

$$\boldsymbol{m} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{\delta} = \begin{bmatrix} 0 & 1 & 1 & 2 & 1 \\ 0 & 1 & 2 & 3 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix} \tag{4}$$

A standard recurrent neural network formulation is shown in Equation (5). Here, $\sigma(\cdot)$ is some activation function, e.g. sigmoid, x_t, h_t are the input vector and recurrent hidden state vector, respectively, and W_h, U_h are learned network parameters.

$$\mathbf{h_t} = \sigma\left(\mathbf{W_h}\mathbf{h_{t-1}} + \mathbf{U_h}\mathbf{x_t} + \mathbf{b_h}\right) \tag{5}$$

$$\hat{\mathbf{x}}_t = \mathbf{W_x}\mathbf{h_{t-1}} + \mathbf{b_x} \tag{6}$$

$$\mathbf{x_t^c} = \mathbf{m_t} \odot \mathbf{x_t} + (\mathbf{1} - \mathbf{m_t}) \odot \hat{\mathbf{x}}_\mathbf{t} \tag{7}$$

$$\gamma_t = \exp\left\{-\max\left(0, \mathbf{W}_\gamma \delta_\mathbf{t} + \mathbf{b}_\gamma\right)\right\} \tag{8}$$

$$\mathbf{h_t} = \sigma\left(\mathbf{W_h}\left[\mathbf{h_{t-1}} \odot \gamma_\mathbf{t}\right] + \mathbf{U_h}\left[\mathbf{x_t^c} \circ \mathbf{m_t}\right] + \mathbf{b_h}\right) \tag{9}$$

$$\ell_t = \left\langle \mathbf{m_t}, \mathcal{L}_\mathbf{e}\left(\mathbf{x_t}, \hat{\mathbf{x}}_\mathbf{t}\right)\right\rangle \tag{10}$$

Equations (6) – (10) detail the operations required to perform imputation, based on a modified version of a recurrent neural architecture.

(6): Weight parameter matrices are learned and multiplied with the networks recurrent hidden state (at the previous time step) to estimate a new input vector with no missing information, $\hat{\mathbf{x}}_t$.

(7): A complement vector, $\mathbf{x_t^c}$, is formed by combining both the available feature values with estimated information when the feature value is missing, where $\odot$ represents element-wise multiplication.

(8): In addition to predicting missing information the neural architecture also applies a temporal decay, γ_t, to the recurrent hidden state, based on the time gap since a feature value was measured, i.e. the $\boldsymbol{\delta}$ matrix. The longer the time gap since the feature was measured results in more exponential decay being applied to the corresponding feature.

(9): The final hidden state vector, $\mathbf{h_t}$, is derived via a learned linear combination of *history-based estimation* (captured in the previously described steps) and *feature based estimation* that learns a further weight matrix, $\mathbf{U_h}$, to mix between available feature values at the present time step, where $\circ$ is the concatenation operator. Equation (9) shows the final hidden state vector computed using a vanilla RNN formulation, whereas in practice an LSTM or GRU could be used to make an outcome prediction for the current time step.

(10): Finally, a loss function is applied (e.g. mean absolute error or mean squared error) between the estimated feature values and the true feature values (when available) that allows the network to learn its weight matrix parameters.

3. Sequence Prediction Network

Given input clinical time series $(\mathbf{x_1^c}, \cdot, \mathbf{x_t^c})$, which are imputed through either RITS or forward-filling, a sequence prediction model $f(\cdot)$ maps the input sequence to an output sequence. In sepsis prediction, y_t represents the probability that the patient would have sepsis in six hours at the t-th hour.

$$\hat{y}_1, \cdots, \hat{y}_t = f(\mathbf{x_1^c}, \cdot, \mathbf{x_t^c}) \tag{11}$$

We select temporal convolutional network (TCN) [7] as the sequence prediction model due to its four desirable properties:

First, both the output sequence and each hidden layer have the same length with the input sequence, which is achieved by adding zero padding to subsequent layers. As a result, TCN can output the patient's probability of having sepsis in six hours at any time during the ICU stay.

Second, TCN uses causal convolutions to avoid the leakage from future to the past.

Third, to enable TCN to utilize long history sequences, dilated convolutions are applied. By setting the dilation factor to be an exponential function of the network depth, the receptive fields of hidden layers grow exponentially w.r.t the network depth. Therefore, the receptive fields can be flexibly adjusted by either changing 1) the number of hidden layers; or 2) the kernel size.

Fourth, the model can be trained in parallel and therefore faster than other sequence prediction models such as the recurrent neural network.

4. Utility Loss

We derive an approximation of the utility function defined in [1] and use it as the learning objective. For sepsis detection, the *optimal detection time* t^* is defined as six hours before the onset time. For a true sepsis patient, the prediction model would be rewarded 1 if it can correctly predict sepsis at the optimal detection time. Both too early prediction (earlier than six hours before the onset) or too late prediction (later than six hours before the onset or after the onset) would receive a reward less than 1. The reward (utility) would decrease as the prediction time moves further from the optimal detection time. We refer the reader to [1] for an illustration of the utility curves.

For a non-sepsis patient, there would be a slight penalty for falsely predicting sepsis. There would be no reward or penalty for correctly identifying the patient as non-sepsis.

Specifically, the utility curve can be expressed as follows

$$U_{TP}(t) = -0.05\mathbb{I}[t < t^* - 6] + (t - t^* + 6)/6 \quad (12)$$
$$\mathbb{I}[t^* - 6 \leq t \leq t^*] + (9 - t + t^*)/9\mathbb{I}[t^* < t \leq t^* + 9]$$
$$U_{FN}(t) = 0\mathbb{I}[t < t^*] - 2(t - t^*)/9\mathbb{I}[t^* \leq t \leq t^* + 9] \quad (13)$$
$$U_{FP}(t) = -0.05; \qquad U_{TN}(t) = 0 \quad (14)$$

For each of N patients, given the sepsis label s_n, optimal detection time t_n^*, and predicted probability of sepsis in six hours at the t-th hour during the patient's ICU stay p_{nt}, we define the learning objective as maximizing the approximated utility function:

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \mathbb{I}[s_n = 1]\big(p_{nt}U_{TP}(t) + \quad (15)$$
$$(1 - p_{nt})U_{FN}(t)\big) + \mathbb{I}[s_n = 0]\big(p_{nt}U_{FP}(t) +$$
$$(1 - p_{nt})U_{TN}(t)\big)$$

where p_{nt} is modeled as a TCN and θ is the TCN parameters.

5. Results

The dataset consists of multivariate clinical time series (vital signs, lab measurements) and static variables (demographics, unit admission information) collected from 60,000 patient ICU stays, from three separate hospital systems (A, B & C). 40,338 records from hospitals A & B, were made available as public training and validation data. The remaining patient ICU stays were held back as test data. Data from Hospital C was only available within the hidden test set, hence requiring algorithms to generalize to an unseen hospital system. Full details of the data and cohort are described in [1].

To train the imputation network, we randomly split the dataset into 10 folds of equal size and apply 10-fold *nested* cross validation. Among these 10 folds, 8 folds are used as training set, 1 fold is used as validation set and the remaining 1 fold is used as the test set. When the fold $i(i = 1, \cdots, 9)$ is used as the test set, fold $i - 1$ is used as the validation set. For test fold 0, fold 9 is used as the validation set.

When training the imputation network, we regularize the reconstruction loss with the utility loss to make the imputed values adaptive to the prediction. Both the network architecture and hyperparameters are tuned by optimizing the loss function on the validation set. The resulting LSTM cell contains one hidden layer of size 64 with dropout rate 0.5.

After imputing missing values, we train a TCN to maximize the approximated utility function. The input features consist of 1) imputed feature values output by the imputation network; and 2) missingness indicator variables of the raw vital signs and lab measurements. We use the same train-validation-test split as training the imputation network. After tuning the TCN architecture and hyperparameters, the optimal TCN consists of one hidden layer with 100 kernels, where the size of each kernel is 5.

We compared the following approaches in terms of test normalized utility:

- **RITS**: apply the trained RITS model to test set and compute utility
- **FF-TCN**: train TCN on forward-filled time series and missingness indicator variables
- **RITS-TCN**: train TCN on RITS-imputed time series and missingness indicator variables

In addition to the neural architectures described above, a tree-based XGBoost model was also submitted to a follow-up 'Hackathon' event hosted by the challenge organizers.

- **XGBoost**: the feature set of the XGBoost model was augmented with a number of observations related to variable missingness, including binary features to indicate whether a feature was missing, or had ever been measured, as well as the measurement rate of each feature and time since last non-missing measurement.

For these four approaches, we show their test normalized utility scores of 10-fold cross validation in Table 1.

First, comparing RITS-TCN and RITS, RITS-TCN consistently outperforms RITS on each test fold in achieving higher test utility score. Therefore, it is necessary to train a separate prediction network even though RITS can combine the reconstruction loss (for imputation) and the utility loss (for prediction) in its objective.

Second, comparing RITS-TCN and FF-TCN, RITS-TCN outperforms FF-TCN on 8 of 10 test folds, indicating the advantage of applying RITS to impute missing values instead of simple forward filling.

Third, comparing different folds within each model, the test utility scores have large variance, indicating the necessity of building ensemble models. This is because the ensemble model can reduce variance, leading to an improved utility score on the test set.

Finally, the additional XGBoost model highlights the usefulness of simpler model types.

Table 1. 10-fold cross validation normalized utility scores of RITS, FF-TCN, RITS-TCN, and XGBoost models

Fold	RITS	FF-TCN	RITS-TCN	XGBoost
0	0.373	0.396	0.397	0.412
1	0.315	0.357	0.354	0.405
2	0.395	0.393	0.415	0.411
3	0.352	0.414	0.425	0.445
4	0.366	0.356	0.382	0.404
5	0.361	0.398	0.392	0.403
6	0.336	0.368	0.376	0.392
7	0.387	0.388	0.416	0.437
8	0.371	0.374	0.387	0.422
9	0.401	0.426	0.430	0.429
Mean (Std)	0.366 (0.025)	0.387 (0.022)	0.397 (0.024)	0.416 (0.017)

Our final submission for the PhysioNet challenge was an ensemble of 3 RITS-TCN models trained using test set folds 3, 7 and 9 from Table 1. The ensemble model would predict sepsis if at least one model would predict sepsis.

The overall utility score achieved by the model was *0.328*. Table 2 shows utility on the challenge test set broken down by hospital. The subsequent XGBoost model developed at the follow-up 'hackathon' event, held by the challenge organizers, improved the utility score to *0.342* and achieved an official rank of 2nd place. Table 2 also shows the utility scores achieved by the official challenge winner [8].

Table 2. Final official test utility scores per hospital for our submission to 2019 PhysioNet Computing in Cardiology Challenge and Hackathon (team name = prna). Rankings are per event.

Hospital Set	Challenge	Hackathon	Winner [8]
A	0.411	0.417	0.433
B	0.389	0.42	0.434
C	-0.159	-0.156	-0.123
Full Set	**0.328**	**0.342**	**0.360**
Official Rank	*9th*	*2nd*	*1st*

6. Conclusions and Future Work

In this work, we built a sepsis prediction model by applying RITS to impute missing values in multivariate clinical time series followed by TCN to maximize the approximated utility objective. Experimental results on the PhysioNet Challenge 2019 sepsis cohort demonstrate the effectiveness of our proposed approach. An additional XGBoost model submitted to a follow-up 'hackathon' event, highlighted the effectiveness of simpler model types for this problem.

The diversity of an ensemble model could be improved by increasing the number of RITS-TCN models, as well as including other prediction models types, potentially leading to higher test utility scores.

In future work, we would also like to explore model interpretation through the attention mechanism for blackbox models, such as RITS and TCN, in order to better incorporate these models into clinicians' workflows.

References

[1] Reyna MA, Josef C, Jeter R, Shashikumar SP, M. Brandon Westover MB, Nemati S, Clifford GD, Sharma A. Early Prediction of Sepsis from Clinical Data: the PhysioNet/Computing in Cardiology Challenge 2019. Critical Care Medicine 2019;In press.

[2] https://www.cdc.gov/sepsis/datareports/index.html.

[3] Nemati S, Holder A, Razmi F, Stanley MD, Clifford GD, Buchman TG. An Interpretable Machine Learning Model for Accurate Prediction of Sepsis in the ICU. Critical care medicine 2018;46(4):547–553.

[4] Barton C, Chettipally U, Zhou Y, Jiang Z, Lynn-Palevsky A, Le S, Calvert J, Das R. Evaluation of a Machine Learning Algorithm for up to 48-hour Advance Prediction of Sepsis using Six Vital Signs. Computers in biology and medicine 2019;109:79–84.

[5] Futoma J, Hariharan S, Sendak M, Brajer N, Clement M, Bedoya A, O'Brien C, Heller K. An Improved Multi-Output Gaussian Process RNN with Real-Time Validation for Early Sepsis Detection. 2nd Machine Learning for Healthcare Conference MLHC 2017;.

[6] Cao W, Wang D, Li J, Zhou H, Li L, Li Y. BRITS: Bidirectional Recurrent Imputation for Time Series. In Advances in Neural Information Processing Systems. 2018; 6775–6785.

[7] Bai S, Kolter JZ, Koltun V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv preprint arXiv180301271 2018;.

[8] Morrill J, Kormilitzin A, Nevado-Holgado A, Swaminathan S, Howison S, Lyons T. The Signature-based Model for Early Detection of Sepsis from Electronic Health Records in the Intensive Care Unit. In 2019 Computing in Cardiology Conference (CinC). IEEE, 2019; (To Appear).

Address for correspondence:

Yale Chang / Jonathan Rubin
2 Canal Park, 3rd floor, Cambridge, MA 02141, USA
yale.chang@philips.com / jonathan.rubin@philips.com