Hybrid Feature Learning Using Autoencoders for Early Prediction of Sepsis

Jia Yao, Ming Lun Ong, Kar Kin Mun, Shiyu Liu, and Mehul Motani

Department of Electrical and Computer Engineering
National University of Singapore, Singapore

Abstract

The early prediction of sepsis is important for ICU patients, as the risk of mortality increases as the disease is left untreated. We hypothesize that there is a need to learn important feature representations, such as to extract salient information from sepsis data. In this paper, we propose an unsupervised method to learn spatial-temporal information from the data, through the use of two autoencoders. For the official 2019 PhysioNet Challenge, our team, Kent Ridge AI (ranked 77th), obtained a utility score of -0.164 on the full test set. Additionally, we report cross-validation results and identify several issues which can potentially help to improve performance.

1. Introduction

Intensive-care unit (ICU) patients are at a high risk of developing sepsis, significantly increasing their risk of mortality[1, 2]. Sepsis causes organ failure and is hard to detect early, motivating the early detection of the disease. In this paper, we propose a hybrid feature learning model, comprising of spatial and temporal autoencoders, to learn deep feature representations. Underlying this decision is our hypothesis that patient data provided contains a great amount of spatial and temporal information. In our model, we stack spatial and temporal autoencoders, such that the hybrid model is able to identify patterns in the two domains. Thus, we can learn deep feature representations from patient data, for the better diagnosis of sepsis.

2. Spatio-temporal Feature Learning

Our proposed hybrid spatio-temporal model consists of a spatial autoencoder (SAE) and a temporal autoencoder (TAE), to learn from both the spatial and temporal domains. In this paper, $X \in \mathbb{R}^{\mathbb{S}\times\mathbb{T}}$ is used to represent the raw patient data, where S is the number of channels and T is the number of time stamps. Therefore, we consider X in the form, such as $X = [x_1, x_2, ..., x_\tau]$, where $x_\tau \in \mathbb{R}^{\mathbb{S}}, \forall\tau = 1, ..., \mathrm{T}$. x_τ includes all channel records at time stamp τ.

2.1. Spatial Autoencoder (SAE)

The SAE is a standard autoencoder constructed with a multi-layer neural network [3] [4], consisting of an encoder and a decoder. The encoder, Spatial-E, has multiple hidden layers, where each successive layer extracts a compact representation of the previous hidden layer. Through the Spatial-E encoder, the input data at each time stamp $x_\tau \in \mathbb{R}^{\mathbb{S}}$ is mapped into a compact representation $Y_\tau \in \mathbb{R}^{\mathbb{S}'}$ through a deterministic mapping:

$$Y_\tau = f_{\boldsymbol{\theta}^{se}}(x_\tau) = s(\mathbf{W}\mathbf{x}_\tau + \mathbf{b}), \tag{1}$$

parameterized by $\boldsymbol{\theta}^{\mathbf{se}} = (\mathbf{W}, \mathbf{b})$, with the weight matrix $\mathbf{W} \in \mathbb{R}^{\mathbb{S}'\times\mathbb{S}}$ and the bias vector $\mathbf{b} \in \mathbb{R}^{\mathbb{S}'}$. $s(.)$ denotes an activation function, performing a non-linear transformation of the given data. The spatial encoder, denoted by $\boldsymbol{Y} = [Y_1, Y_2, ..., Y_\tau]$, where $Y_\tau \in \mathbb{R}^{\mathbb{S}'}, \forall\tau = 1, ..., \mathrm{T}$, represents spatial representations of the raw input data X. The decoder, Spatial-D, has the same symmetric structure as the encoder Spatial-E. The compact representations generated by the encoder Spatial-E at each time stamp τ (i.e., $Y_\tau \in \mathbb{R}^{\mathbb{S}'}$) are reconstructed back to the input data $\hat{x}_\tau \in \mathbb{R}^{\mathbb{S}}$ through a deterministic mapping:

$$\hat{x}_\tau = f_{\boldsymbol{\theta}^{se'}}(Y_\tau) = s(\mathbf{W}'\mathbf{Y}_\tau + \mathbf{b}'), \tag{2}$$

which parameterized by $\boldsymbol{\theta}^{\mathbf{se}'} = (\mathbf{W}', \mathbf{b}')$, where $\mathbf{W}' \in \mathbb{R}^{\mathbb{S}\times\mathbb{S}'}$ and $\mathbf{b}' \in \mathbb{R}^{\mathbb{S}}$. To train the SAE, the multivariate time series patient data X is first sliced into x_τ, (i.e., x_τ is the patient's multi-channel measurements recorded at time stamp τ), where $\tau = 1, ..., T$. Each x_τ is used to train the SAE, and update the parameters $\boldsymbol{\theta}^{se}$ and $\boldsymbol{\theta}^{se'}$. In our study, P number of patients data are involved for training. Each patient has a length of T measurements recorded in total. Therefore, the SAE is trained by $P \times T$ number of data samples iteratively. The rectified linear unit and sigmoid are used as the activation functions for the Spatial-E and Spatial-D, respectively. Both parameters $\boldsymbol{\theta}^{se}$ and $\boldsymbol{\theta}^{se'}$ are updated and trained, such as to minimize the average reconstruction error between x_τ and $\hat{x}_\tau$ [4], [5], measured by the mean squared error (MSE) loss function $L(x_\tau, \hat{x}_\tau)$. The SAE is optimized using Adam [6].

 ISSN: 2325-887X DOI: 10.22489/CinC.2019.243

2.2. Temporal Autoencoder (TAE)

The TAE in this paper is constructed with the long short-term memory (LSTM) cell [7] a type of recurrent neural network (RNN). Each LSTM cell contains four gates: the forget gate (f), input gate (i), update gate (u) and output gate (o). Each gate takes in both the current input (e.g., x_τ) and the output from the hidden state (e.g., $h_{\tau-1}$) of the previous LSTM cell. The mathematical operations within each LSTM cell at time stamp τ are computed based on the following:

$$f_\tau = A_f(W_f[h_{\tau-1}, x_\tau] + b_f) \quad (3)$$

$$i_\tau = A_i(W_i[h_{\tau-1}, x_\tau] + b_i) \quad (4)$$

$$u_\tau = A_u(W_u[h_{\tau-1}, x_\tau] + b_u) \quad (5)$$

$$C_\tau = C_{\tau-1} * f_\tau + i_\tau * u_\tau \quad (6)$$

$$o_\tau = A_o(W_o[h_{\tau-1}, x_\tau] + b_o) \quad (7)$$

$$h_\tau = o_\tau * \tanh(C_\tau). \quad (8)$$

Each gate in an LSTM-E encoder cell of TAE has one hidden layer with a total number of D units, which are used to learn and extract temporal dependencies from the multivariate time series input data X. The dimension is $S \times T$, where S is the number of channel measurements and T is the length of time stamps. As a result, (3)-(8), $f_\tau \in \mathbb{R}^{\mathbb{D}}$, $i_\tau \in \mathbb{R}^{\mathbb{D}}$, $u_\tau \in \mathbb{R}^{\mathbb{D}}$ and $o_\tau \in \mathbb{R}^{\mathbb{D}}$ are the output signals from the four gates, and A_f, A_i, A_u and A_o are the activation functions at each gate respectively. The sigmoid function is used for A_f, A_i and A_o, and the hyperbolic function is used for A_u. The cell state ($C_\tau \in \mathbb{R}^{\mathbb{D}}$) of the current LSTM cell is updated based on the previous cell state, and controlled by the outputs from the forget gate, input gate and update gate, whereas the hidden state ($h_\tau \in \mathbb{R}^{\mathbb{D}}$) of the current LSTM cell is computed by the signal from the output gate and the activated output of the current cell state. Then, the combined signals, C_τ and h_τ, are the ones which will be passed to the next LSTM cell corresponding to the time stamp $\tau + 1$. $\boldsymbol{\theta}^{te} = \{W_f, b_f, W_i, b_i, W_u, b_u, W_o, b_o\}$ are the parameters for each LSTM-E. In this paper, we assume that the LSTM-E at different time stamps share the same set of parameters.

Figure 1 illustrates the structure of TAE. We take outputs from the hidden states of every encoder cell LSTM-E in the TAE encoder, and consider them as the temporal feature representations $\boldsymbol{h} \in \mathbb{R}^{\mathbb{D}\times\mathbb{T}}$ of the input data X[8] [9]. In order to train the TAE, a decoder consisting of a set of LSTM-D decoder cells, each with S hidden units in all the gates, are attached after the encoder, mapping the encoded temporal feature representations $\boldsymbol{h}$ back to the input of TAE, X. Having the same setting as the TAE encoder, all the LSTM-D in the decoder share the same set of parameters, denoted by $\boldsymbol{\theta}^{te'}$.

During training, the LSTM-E encoder takes in the raw data $X \in \mathbb{R}^{\mathbb{S}\times\mathbb{T}}$ to capture the temporal variations. As

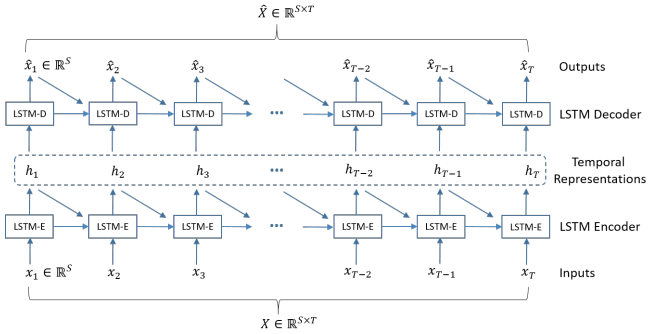

Figure 1. Temporal autoencoder.

a result, the outputs from the hidden states of all the LSTM cells in the encoder of TAE, i.e., $\boldsymbol{h} \in \mathbb{R}^{\mathbb{D}\times\mathbb{T}}$, and $\boldsymbol{h} = [h_1, h_2, ..., h_T]$ is then fed into the decoder of TAE i.e., LSTM-D, to reconstruct the raw data $\hat{X}$ by decoding, where $\hat{X} \in \mathbb{R}^{\mathbb{S}\times\mathbb{T}}$ is the reconstructed signal from the TAE decoder based on $\boldsymbol{h}$. The parameters of the TAE (e.g., $\boldsymbol{\theta}^{te}$) are optimized by minimizing the MSE reconstruction error via Adam. Similar to the spatial autoencoder, we use P patients in the training process. Thus, the parameters of TAE are iteratively updated and trained with P data samples.

2.3. Stacking Spatial and Temporal Autoencoders

As motivated by [10], the two types of autoencoders described above can be stacked together to learn deeper feature representations from both spatial and temporal domains. We use STAE to represent the stacked autoencoders, where a TAE is attached after a SAE (see Figure 2). In the SAE model, the data X is passed into a SAE to learn spatial representations denoted by $\boldsymbol{Y}$, and then $\boldsymbol{Y}$ is passed into a TAE in the next stage to learn temporal representations, denoted by $\boldsymbol{h}$. The autoencoder at each stage of the model is trained and optimized individually. Similar to STAE model, we use TSAE to denote a model which stacks a TAE and an SAE. In this TSAE architecture, the temporal representation $\boldsymbol{h}$ generated by TAE is passed into a SAE to further learn the spatial representation $\boldsymbol{Y}$. To examine the effectiveness of such hybrid stacked autoencoder models in learning feature representations, single-type autoencoder models (e.g., SAE and TAE) are also constructed as baselines for comparison.

3. Performance Evaluation

Officially, our team, Kent Ridge AI (ranked 77th), obtained a utility score of -0.164 on the full test set for the 2019 PhysioNet Challenge. The following section outlines our methods, and intermediate training performance.

Dataset: Within the PhysioNet 2019 Challenge dataset

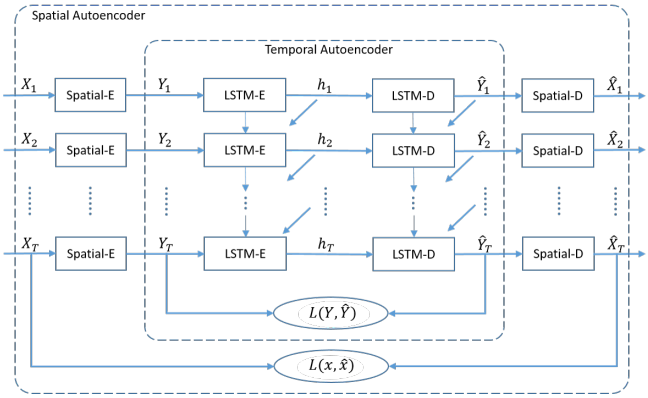

Figure 2. STAE model structure.

[11], we note that many features (e.g., SaO2) contain more than 90% of values which are missing. For these, we impute with their corresponding latest historical values. Additionally, to perform earlier prediction at 12 hours, we shift the sepsis label ahead by another 6 hours. Finally, we normalize the dataset to be within the range 0 to 1, and resample using a window of length 2. We note that during evaluation, we may encounter the case where only a single timestamp is given; for such a case, we double the given timestamp and reshape it into a new size of length 2.

Experimental Setup: The dataset is randomly partitioned into 5-folds on the patient-level, such that 80% of data is used for training, and 20% is used for evaluation. Three benchmark models, the Decision Tree [12], Random Forest [13], Logistic Regression [14] are shortlisted; through this, we show the effect that our architecture provides. The Spatial-E and Spatial-D contain only one hidden layer with a total of $\mathbb{P}$ and $\mathbb{Q}$ neurons respectively. We note that $\mathbb{P}$ determines the output size of SAE and set it to a size of 10. Similarly, Both LSTM-E and LSTM-D contain only one hidden layer with a total of $\mathbb{M}$ and $\mathbb{N}$ neurons, respectively. We note that $\mathbb{M}$ determines the output size of TAE, and we set to be 20. Additionally, the value of $\mathbb{Q}$ and $\mathbb{N}$ depends on the input size of SAE and TAE respectively.

Experimental Results: In this section, we report intermediate training performance, in the form of accuracy, F1 Scores and AUC-ROC values averaged over 10 runs. These intermediate training results are displayed in Table 1. With the random forest model, the results learnt by the TSAE architecture achieves the highest training classification accuracy among all feature sets learned by different feature learning methods, and it also outperforms the raw data. Using the TSAE feature set, we report an intermediate training accuracy of 77.2%, which is about 5.8% and 2.8% higher than using the feature sets extracted from the single-type autoencoder model, i.e., SAE and TAE models for classification respectively. Similar observations can also be seen with decision tree and logistic regression models. Moreover, the AUC-ROC and F1-score achieved by TSAE feature set are generally higher than feature sets learned by other baseline feature learning models and the raw data with all three different classifiers. Moreover, we test the TSAE model on a subset of the hidden test data in one of the submission rounds in the PhysioNet Challenge, resulting in an intermediate training score of -0.046. For the official 2019 PhysioNet Challenge, our team (Kent Ridge AI) ranked 77th and obtained a utility score of -0.164 on the full test set.

3.1. Analysis of STAE and TSAE

The TSAE model outperforms the SAE model because the temporal autoencoder in TSAE can also capture spatial correlations of the raw data. When $x_\tau \in \mathbb{R}^{\mathbb{S}}$ and $h_{\tau-1} \in \mathbb{R}^{\mathbb{D}}$ are passed into the gates in LSTM, the total number of inputs are summarized and re-represented as D hidden units in the network. Therefore, the output from the hidden state of a LSTM cell at time stamp τ, $h_\tau \in \mathbb{R}^{\mathbb{D}}$ demonstrates some learnt spatial dependencies in x_τ. Thus, the structure of TSAE is similar to stacking one temporal autoencoder and two spatial autoencoders together (i.e., one strong and one weak spatial autoencoders). This structure has more capability to capture spatial correlations. Moreover, the temporal autoencoder in TSAE captures important temporal information. Therefore, TSAE has a better classification performance than SAE feature sets.

This rationale also applies to the comparison between TSAE and TAE. The spatial autoencoder in TSAE captures spatial correlations, and the temporal autoencoder has ability to learn certain spatial correlations at the same time, whereas TAE only has limited capabilities to learn spatial correlations. The TSAE is more effective in learning useful feature representations from both spatial and temporal domains, as compared to a structure only constructed by a single-type autoencoder.

While combining the SAE model and TAE model to construct a hybrid feature learning model (i.e., STAE and TSAE), the order of stacked autoencoders is important, as is affects the classification performance of the extracted feature sets. With the PhysioNet Computing in Cardiology Challenge dataset, the feature set extracted by TSAE model is better than the feature set extracted by the STAE model. This is demonstrated in the decision tree model. A similar pattern can also be seen in AUC-ROC and F1-scores in all three classifiers. In this case, we suspect that the order of stacking different types of the autoencoders is contingent on the dataset used, which strongly depends on the amount of spatial and temporal information in the data. For this dataset, we compute correlations, obtaining a spatial correlation value of 0.37 and a temporal correlation

	Decision Tree			Random Forest			Logistic Regression		
	Accuracy	F1	AUC-ROC	Accuracy	F1	AUC-ROC	Accuracy	F1	AUC-ROC
No Autoencoder	0.659	0.236	0.529	0.741	0.153	0.531	0.645	0.247	0.511
SAE	0.668	0.244	0.515	0.730	0.166	0.522	0.597	0.277	0.541
TAE	**0.679**	0.264	**0.541**	0.751	0.173	0.511	**0.604**	0.285	0.534
STAE	0.668	0.268	0.527	0.743	0.168	0.509	0.602	0.283	0.544
TSAE	0.674	**0.276**	0.525	**0.772**	**0.192**	**0.533**	0.593	**0.313**	**0.566**

Table 1. Intermediate training performance obtained from using various configurations of autoencoders

value of 0.95. As the temporal value is much higher, we find that it is more effective to first extract temporal dependency in the data. Conversely, learning spatial information first would potentially destroy the initial strong temporal dependencies in the data.

4. Reflections

In this paper, we suggest using a spatio-temporal autoencoder to better predict sepsis at an early stage, which had demonstrated promising results on other datasets [15]. However, we acknowledge that our method had not been successful in the PhysioNet Challenge, and suspect that the poor performance was linked to our data pre-processing methods. To improve the predictive performance of our model, we propose the use of feature selection methods on the raw dataset to select a subset of relevant features [16]. Additionally, capturing additional information in the temporal domain will be helpful to capture the change in feature value progression over time. For this, we suggest the use of trend features, where we take the difference between feature values from a past time point, and feature values from the next time point.

Acknowledgments

This research work was supported by the Ministry of Education, Singapore, under the grants WBS R-263-000-D35-114 and WBS R-263-000-D64-114.

References

[1] Taylor RA, Pare JR, Venkatesh AK, Mowafi H, Melnick ER, Fleischman W, Hall MK. Prediction of in-hospital mortality in emergency department patients with sepsis: a local big data–driven, machine learning approach. Academic emergency medicine 2016;23(3):269–278.

[2] Desautels T, Calvert J, Hoffman J, Jay M, Kerem Y, Shieh L, Shimabukuro D, Chettipally U, Feldman MD, Barton C, et al. Prediction of sepsis in the intensive care unit with minimal electronic health record data: a machine learning approach. JMIR medical informatics 2016;4(3):e28.

[3] Vincent P, Larochelle H, Bengio Y, Manzagol PA. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, ICML '08. ISBN 978-1-60558-205-4, 2008; 1096–1103.

[4] Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res ;11.

[5] Bell AJ, Sejnowski TJ. An information-maximization approach to blind separation and blind deconvolution. Neural Comput November 1995;7(6).

[6] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv14126980 2014;.

[7] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput ;9.

[8] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems (NIPS). 2014; .

[9] Cho K, van Merrienboer B, Bahdanau D, Bengio Y. On the properties of neural machine translation: Encoder-decoder approaches. In Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014. 2014; .

[10] Patraucean V, Handa A, Cipolla R. Spatio-temporal video autoencoder with differentiable memory. CoRR 2015;.

[11] 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.

[12] Quinlan JR. Induction of decision trees. Mach Learn 1986; ISSN 0885-6125.

[13] Breiman L. Random forests. Machine Learning 2001;45.

[14] Wright RE. Logistic regressions. Reading and understanding multivariate statistics 2004;217–244.

[15] Yao J, Motani M. Deep spatio-temporal feature learning using autoencoders. Workshop on Modeling and Decision Making in the Spatiotemporal Domain at NeurIPS 2018 2018;URL https://openreview.net/pdf?id=r1gjEXnAK7.

[16] Liu S, Yao J, Zhou C, Motani M. Suri: Feature selection based on unique relevant information for health data. In 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE, 2018; 687–692.

Address for correspondence:

Jia Yao
4 Engineering Drive 3, E4-06-12,
Communication Lab, Singapore 117583
yao.jia@u.nus.edu