Modeling Alzheimer's disease progression using deep recurrent neural networks

Minh Nguyen[1,2], Nanbo Sun[1,2], Daniel C. Alexander[3], Jiashi Feng[2], B.T. Thomas Yeo[1,2,4,5,6] for the Alzheimer's Disease Neuroimaging Initiative[*]
[1] Clinical Imaging Research Centre Singapore Institute for Neurotechnology, National University of Singapore, Singapore
[2] Department of Electrical and Computer Engineering, National University of Singapore, Singapore
[3] Centre for Medical Image Computing, Department of Computer Science, University College London, London, UK
[4] Martinos Center for Biomedical Imaging, Massachusetts General Hospital, Charlestown, MA, USA
[5] Centre for Cognitive Neuroscience, Duke-NUS Medical School, Singapore
[6] NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, Singapore

Abstract — **Multi-step prediction, predicting values multiple steps ahead, is a difficult problem because of the accumulation of error after successive predictions. Recurrent neural networks (RNN) have been applied successfully to multi-step prediction problems such as language modeling where training data are abundant and regular. However, applying RNN to problems in healthcare is still challenging as RNN requires large, feature complete training data in which intervals between time points are fixed which are not reflective of healthcare data. To overcome these problems, we propose strategies to train RNN using missing data which involve using the RNN to fill in the incomplete or missing time points online. The strategies enables the efficient use of training data as well as the adoption of RNN in problems where the variables in a multivariate time series are recorded at irregular intervals or at different frequencies, where there are a lot of missing data. We train RNN using the Alzheimer's Disease Neuroimaging Initiative (ADNI) database to model the monthly progression of Alzheimer's disease for seven years into the future which is a multi-step prediction problem. We show that RNN outperforms baseline models.**

Keywords — disease progression modeling, multi-step prediction, recurrent neural networks, missing data imputation

I. Introduction

Alzheimer's disease (AD) is a devastating neurodegenerative disease. There is significant interest in predicting AD progression [1, 2, 3] which would be clinically useful for improving diagnosis, monitoring disease progression and identifying individuals for clinical trials [4]. Most of prior works only consider the problem of single-step prediction, i.e., predicting the variables of interest once at a fixed time horizon in the future. In this paper, we consider the harder multi-step prediction problem where the models need to predict multiple disease markers of an individual every month, up to seven years into the future.

Proposed approaches for multi-step prediction include independent prediction (whereby future values are predicted independently) and multi-stage prediction (whereby next values are predicted based on current values) [5]. Multi-stage prediction is prone to accumulation of error, leading to inaccurate distant predictions while the output of independent prediction may lack temporal coherence. Besides, the presence of missing data, i.e., data entries with missing time points or missing features, in healthcare datasets also makes multi-step disease progression modeling difficult as predictive models are usually trained using complete data. Missing data arises due to various reasons such as the nature of data collection procedures, subjects' dropping out of studies or mistakes in data collection. For example, the ADNI protocol dictates that subjects from different categories undergo different tests at different time points.

RNN can be used for multi-step prediction following the multi-stage prediction approach. RNN have been applied to single-step prediction problems in healthcare [6, 7, 8, 9]. However, the adoption of RNN in healthcare is still rare due to many problems. As healthcare datasets are expensive to collect, they are often small in size. Without sufficient training data, RNN overfit easily and do not generalize as well as simple linear models [10]. Besides, RNN need complete data for training so missing data is discarded making datasets even smaller. In addition, RNN require fixed durations between time points. This assumption is usually invalid in healthcare data since different tests are conducted at different frequencies.

We propose strategies to apply RNN the multi-step prediction problem of modeling AD progression such that the RNN can be trained using missing data, which is common in real-world data, but is mostly ignored in the literature [1, 2]. One of the strategies involves using the RNN itself to interpolate and extrapolate the missing data during training, enabling efficient use of training data. The strategies are also used to fill in data between time points, allowing the duration between model to be fixed but much shorter than duration between time points. Our result shows RNN outperform baseline models.

II. Method

A. Recurrent Neural Networks

The RNN architecture is shown in **Figure 1**. The Long Short-Term Memory variant [11] of RNN is used because it is more effective in modeling long sequences. Given a subject's state at time t, the RNN predicts the subject state at time $t+1$. The state at time t includes a categorical variable and multiple continuous variables. The categorical variable is the diagnosis of the subject at time t, i.e., whether the subject is normal control (NC), mild cognitive impairment (MCI) or AD patient. The continuous variables include potential AD markers such as

*Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf

ADAS-Cog13 and MMSE score, ventricles and hippocampal volume, CSF amyloid and tau. The choice of markers and the design of the model were inspired by the TADPOLE challenge [12] which aims to identify people at risk of developing AD symptoms within five years.

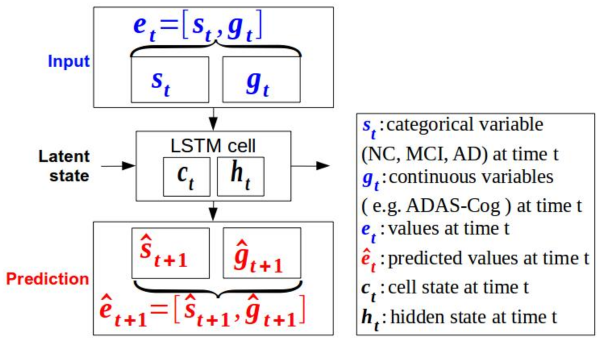
Input
$e_t=[s_t, g_t]$
s_t
g_t
Latent state
LSTM cell
c_t
h_t
Prediction
$\hat{s}_{t+1}$
$\hat{g}_{t+1}$
$\hat{e}_{t+1}=[\hat{s}_{t+1}, \hat{g}_{t+1}]$
s_t: categorical variable (NC, MCI, AD) at time t
g_t: continuous variables (e.g. ADAS-Cog) at time t
e_t: values at time t
$\hat{e}_t$: predicted values at time t
c_t: cell state at time t
h_t: hidden state at time t

Figure 1: Model architecture

Specifically, let $\boldsymbol{s}_t$, and $\boldsymbol{g}_t$ be vectors containing values of the diagnosis and AD markers at time t of a subject respectively. The input to the RNN at time t is $\boldsymbol{e}_t$ which is the concatenation of vectors $\boldsymbol{s}_t$ and $\boldsymbol{g}_t$ (Equation 1). Given the subject's present state $\boldsymbol{e}_t$, the RNN updates the hidden vector $\boldsymbol{h}_t$ and cell vector $\boldsymbol{c}_t$ (Equations 2 - 7) and predict the subject's future state $\hat{\boldsymbol{e}}_{t+1}$.

$$\boldsymbol{e}_t = [\, \boldsymbol{s}_t, \boldsymbol{g}_t \,] \tag{1}$$

$$\boldsymbol{i}_t = \sigma\,(\, \boldsymbol{W}^{(i)}\, \boldsymbol{e}_t + \boldsymbol{U}^{(i)}\, \boldsymbol{h}_{t-1} \,) \tag{2}$$
$$\boldsymbol{f}_t = \sigma\,(\, \boldsymbol{W}^{(f)}\, \boldsymbol{e}_t + \boldsymbol{U}^{(f)}\, \boldsymbol{h}_{t-1} \,) \tag{3}$$
$$\boldsymbol{o}_t = \sigma\,(\, \boldsymbol{W}^{(o)}\, \boldsymbol{e}_t + \boldsymbol{U}^{(o)}\, \boldsymbol{h}_{t-1} \,) \tag{4}$$
$$\boldsymbol{u}_t = \tanh\,(\, \boldsymbol{W}^{(u)}\, \boldsymbol{e}_t + \boldsymbol{U}^{(u)}\, \boldsymbol{h}_{t-1} \,) \tag{5}$$

$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \boldsymbol{u}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \tag{6}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh\,(\, \boldsymbol{c}_t \,) \tag{7}$$

$$\hat{\boldsymbol{s}}_t = \text{softmax}\,(\, \boldsymbol{W}^{(s)}\, \boldsymbol{h}_t \,) \tag{8}$$
$$\hat{\boldsymbol{g}}_t = \text{hardtanh}\,(\, \boldsymbol{W}^{(g)}\, \boldsymbol{h}_t \,) \tag{9}$$

$$\text{hardtanh}\,(\circ) = \min\,(\, \max\,(\circ, 0\,), 1\,) \tag{10}$$
$$\hat{\boldsymbol{e}}_t = [\, \hat{\boldsymbol{s}}_t, \hat{\boldsymbol{g}}_t \,] \tag{11}$$

The σ symbol represents the sigmoid function. The ʘ symbol represents the element-wise product. $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$ are the input gate, forget gate, output gate of the RNN. These gates form the gating mechanism of the RNN, controlling the information stored in the hidden vector [11].

For subjects in the ADNI study, the duration between visits is six months. However, some subjects did not turn up at the designated examination date so the actual duration between time points varies. As RNN requires the duration between visits to be uniform, one could assign the time of the visits to the nearest sixth month and model changes after every six months. However, this jitter in sampling may affect modeling accuracy. Instead, we model the changes with a temporal resolution of one month, i.e., the duration between t and $t+1$ is one month so that there is less sampling jitter. This, however, results in a lot of time points with missing observations.

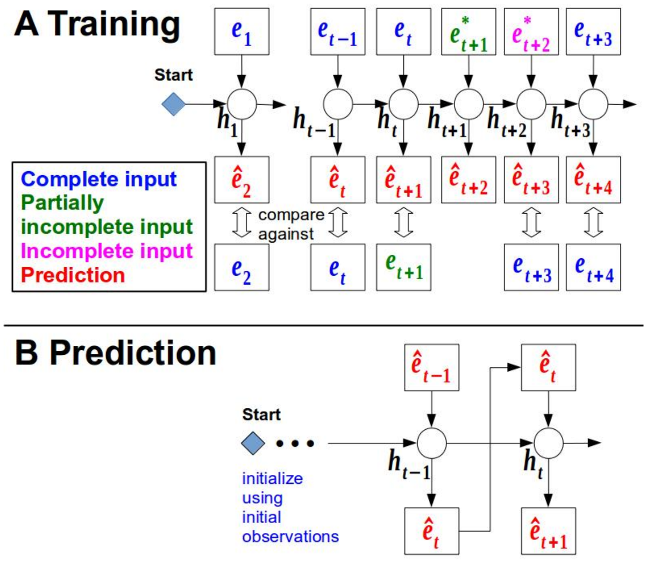

Figure 2: Training and prediction

The training strategy with missing data is shown in **Figure 2A**. For time points such as $t+1$ where only some features are missing, only the missing values are imputed. In the figure, $\boldsymbol{e}^*_{t+1}$ denotes the imputed input while $\boldsymbol{e}_{t+1}$ denotes the original input with missing features. For missing time points such as $t+2$, all the features are imputed. **Section 2B** outlines how data imputation is done. During training, gradients of the errors between predicted and true values are back-propagated to update the RNN parameters. The error (loss) function is only evaluated at time points where observations are available (e.g., $\boldsymbol{e}_t$, $\boldsymbol{e}_{t+3}$, $\boldsymbol{e}_{t+4}$) and not at time points where observations are imputed (e.g., $\boldsymbol{e}^*_{t+2}$).

Figure 2B shows how the RNN is used to predict the progression of Alzheimer's disease in a subject. The RNN makes prediction of the next observations $\hat{\boldsymbol{e}}_t$ based on the current observations $\boldsymbol{e}_t$ and use this prediction as input observations as the subsequent step (i.e., $\boldsymbol{e}_{t+1} = \hat{\boldsymbol{e}}_t$).

B. Imputing missing data

There are many variants to the longitudinal data imputation in **Section 2A**. In this paper, we explore two of the variants namely forward-filling and model-filling.

1) Forward-filling

Figure 3A shows how forward-filling in time is used to fill in missing input features. This method was originally proposed in [8] for single-step classification. In this example, there are two input features A and B. The values of feature A at time t = 2, 3, 4 is filled using the last observed value of feature A (at time t = 1). Similarly, the value at t = 7, 8 of feature A is filled using value at t = 6 when it was last observed.

2) Model-filling

Figure 3B shows the RNN model is used together with linear interpolation to fill in missing data. For time points where there is at least one feature with ground truth data, such as when t = 3, the value of feature A is filled using linear interpolation. At time t = 2, 4, 6, the values of feature A is filled using the prediction from the recurrent neural network. For example, the predicted value of feature A at t = 6, obtained from the RNN using input features at t = 5, is used as input to predict values at t = 7. The RNN can be used to extrapolate features that "terminate early". At time t = 9, the RNN is used to extrapolate the missing data. This maximizes the use of available data.

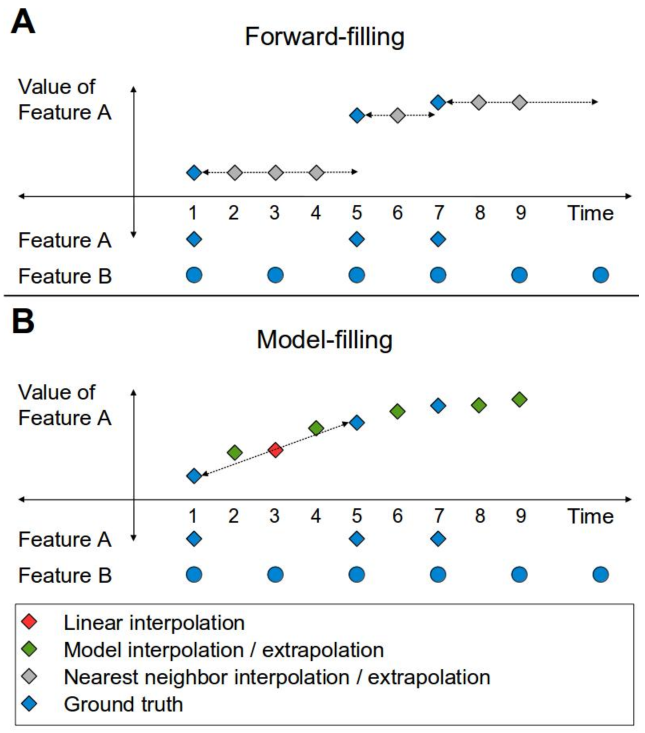

Figure 3: Handling missing data

III. Experiment

A. Data and Setup

We used the TADPOLE dataset which is derived from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.lonu.usc.edu). We generated ten random splits of the data. Each split consists of a training set, a development set, and a test set. For each random split, grid-search was used for hyper-parameter search using result on the development set. Each training set has about 1500 subjects while each development set and test set has about 100 subjects. Each subject has from 1 to 7 time points which are roughly 6 months apart. Values of continuous variables are normalized to [0, 1] range. Value of the categorical variable at each time point can be one of NC, MCI and AD. The extent of missing data varies from variables to variables and ranges from 30% of data missing for the best variable to 80% of the data missing for the worst variable [12].

B. Metrics

We evaluated the models based on the diagnosis classification accuracy and ADAS-Cog13 score and ventricles volume regression accuracy. Classification accuracy is evaluated using multiclass area under the operating curve (mAUC; higher the better) and balanced class accuracy (BCA; higher the better). Regression accuracy is evaluated using mean absolute error (MAE; lower the better). There is a separate set of hyper-parameters for each of the metrics.

C. RNN model

The RNN model is implemented using Pytorch [1]. It has one hidden layer of size 256. The bias term of the forget gate of the RNN is initialized to a large positive value similar to [13]. The other model parameters are initialized to small random values. Variational dropout [10] and L2 weight regularization are used to prevent overfitting. The RNN model is trained using ADAM [14] for 100 epochs. The loss function is the unweighted sum of cross-entropy loss of the categorical variable and MAE loss of the continuous variables.

RNN hyper-parameters are shown in **Table I**. For each set of hyper-parameters, we train ten RNN using ten random seeds and pick five seeds which result in the lowest training error. The RNN models of the top five random seeds produce five different predictions for each subject in the development set. The predictions are averaged and then evaluated against the ground truth. This is to reduce the variance of the prediction as well as to ensure the model performance is robust against random initialization.

TABLE I. RNN and LSS's Hyper-parameters

Model	RNN	LSS
Learning rate	1e-3	1e-5
No. hidden layers	1	
Hidden layer size	256	
Input dropout	0 or 0.1 or 0.2	
Recurrent dropout	0 or 0.2 or 0.5	
L2 regularization	0 or 1e-6 or 1e-5	

After picking the best hyper-parameters, the RNN model is retrained using data from both the training set and the development set. Ten different random seeds are used and the predictions of the top five seeds are averaged. The averaged prediction is evaluated against the ground truth.

The RNN is paired with imputation strategies described in **Section 2B**, resulting in two variants of the same model. The RNN with forward-filling is referred to as RNN-FF while the RNN with model-filling is referred to as RNN-MF. The size of the models are limited at 1 recurrent of size 256 memory cells to prevent overfitting due to the limited amount of training data.

D. Baselines

The RNN were compared against linear baselines such as constant prediction model, Support-Vector Machine (SVM), Support-Vector Regression (SVR) and linear state-space (LSS) model. The input features of the RNN and the baseline models are the same.

1) Constant Prediction Model

The constant prediction model predicts all future values to be the last observed values. This model does not need training.

[1] http://pytorch.org

2) SVM-SVR Model

SVM is used to predict future diagnosis while SVR is used to predict future AD markers. Adopting the independent prediction strategy, 84 SVM and 84 SVR models are trained for 84 time horizons. Linear interpolation is used to impute missing features in the training data. The models are implemented using scikit-learn[1] library. The possible hyper-parameters are shown in **Table II**.

TABLE II. SVM AND SVR'S HYPER-PARAMETERS

	SVM	SVR
Kernel	Linear or RBF	
Epsilon	NA	1e-3, 1e-2, 1e-1
Penalty	1e-2, 1e-1, 1, 1e1, 1e2	
Gamma	1e-2, 1e-1, 1, 1e1, 1e2	

3) Linear State-Space Model

The LSS baseline model was implemented as a simplified RNN model. Equations (12) - (13) show how the LSS update its hidden vector and make prediction. Apart from the update equations, the training scheme, optimizer, regularization scheme of the LSS model are similar to that of the RNN model. The LSS baseline also utilizes the data imputation strategies discussed in section II, resulting in two variants of LSS: LSS-FF and LSS-MF. The hyper-parameters used for the LSS model is shown in **Table I**.

$$\boldsymbol{h}_t = \boldsymbol{W}^{(h)} \boldsymbol{e}_t + \boldsymbol{U}^{(h)} \boldsymbol{h}_{t-1} \tag{12}$$
$$\hat{\boldsymbol{s}}_t = \boldsymbol{W}^{(s)} \boldsymbol{h}_t \tag{13}$$
$$\hat{\boldsymbol{g}}_t = \boldsymbol{W}^{(g)} \boldsymbol{h}_t \tag{14}$$

E. Results

Table III shows the mean and standard deviation of different evaluation metrics on the ten random splits. The RNN with forward-filling strategy (RNN-FF) model achieves the highest accuracy in prediction of diagnosis (mAUC 0.86 and BCA 0.79). The RNN with model-filling strategy (RNN-MF) model achieves the best accuracy for predicting ADAS-Cog13 (MAE 5.2) and ventricles volume (MAE 3.1e-3).

TABLE III. EVALUATION ON TEN DIFFERENT RANDOM SPLITS

Model	ADAS13 MAE	Ventricles MAE (*1e-3)	Diagnosis BCA	Diagnosis mAUC
Baselines				
Constant	5.6 ± 0.31	4.0 ± 0.24	0.68 ± 0.015	0.73 ± 0.016
SVM-SVR	7.6 ± 0.47	10.1 ± 0.97	0.51 ± 0.010	0.55 ± 0.009
LSS-FF	7.9 ± 1.57	4.4 ± 2.05	0.74 ± 0.026	0.85 ± 0.026
LSS-MF	7.6 ± 1.48	10.5 ± 1.32	0.74 ± 0.022	0.82 ± 0.033
Proposed				
RNN-FF	7.2 ± 1.95	11.1 ± 7.50	**0.79 ± 0.035**	**0.86 ± 0.032**
RNN-MF	**5.2 ± 0.45**	**3.1 ± 0.59**	0.74 ± 0.042	0.83 ± 0.035

IV. CONCLUSION

We propose a strategy to use RNN that can handle missing data which is common in healthcare data. We applied it to the multi-step prediction task of modeling AD progression within a seven-year period. The RNN models were more accurate than the baseline models.

ACKNOWLEDGMENT

We are currently supported by Singapore MOE Tier 2 (MOE2014-T2-2-016), NUS Strategic Research (DPRT/944/09/14), NUS SOM Aspiration Fund (R185000271720), Singapore NMRC (CBRG/0088/2015), NUS YIA and the Singapore National Research Foundation (NRF) Fellowship (Class of 2017). Our research also utilized resources provided by the Center for Functional Neuroimaging Technologies, P41EB015896 and instruments supported by 1S10RR023401, 1S10RR019307, and 1S10RR023043 from the Athinoula A. Martinos Center for Biomedical Imaging at the Massachusetts General Hospital. Our computational work was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg). The Titan Xp used for this research was donated by the NVIDIA Corporation. Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012).

REFERENCES

[1] D. Zhang and D. Shen, “Multi-Modal Multi-Task Learning for Joint Prediction of Multiple Regression and Classification Variables in Alzheimer’s Disease,” *NeuroImage*, 2012.

[2] E. Moradi et al., “Machine learning framework for early MRI-based Alzheimer’s conversion prediction in MCI subjects,” *NeuroImage*, 2015.

[3] X. Zhang et al., “Bayesian model reveals latent atrophy factors with dissociable cognitive trajectories in Alzheimer’s disease,” *Proc Natl Acad Sci USA*, 2016.

[4] S. Teipel et al., “Multimodal imaging in Alzheimer’s disease: validity and usefulness for early detection,” *The Lancet Neurology*, Oct. 2015.

[5] H. Cheng et al., “Multistep-Ahead Time Series Prediction,” in *Advances in Knowledge Discovery and Data Mining*, 2006.

[6] Z. C. Lipton et al., “Learning to Diagnose with LSTM Recurrent Neural Networks,” in *Proc Inter Conf on Machine Learning*, 2016.

[7] Z. Che et al., “Recurrent Neural Networks for Multivariate Time Series with Missing Values,” *Scientific reports*, 2018.

[8] H.-G. Kim et al., “Recurrent neural networks with missing information imputation for medical examination data prediction,” in *IEEE Inter. Conf. on Big Data and Smart Computing (BigComp)*, 2017.

[9] Z. C. Lipton et al., “Modeling missing data in clinical time series with rnns,” *Machine Learning for Healthcare*, 2016.

[10] Y. Gal and Z. Ghahramani, “A Theoretically Grounded Application of Dropout in Recurrent Neural Networks,” in *Proc Inter Conf on Neural Information Processing Systems*, 2016.

[11] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, 1997.

[12] R. V. Marinescu et al., "TADPOLE Challenge: Prediction of Longitudinal Evolution in Alzheimer’s Disease," *arXiv:1805.03909*, 2018.

[13] R. Jozefowicz et al., “An Empirical Exploration of Recurrent Network Architectures,” in *Proc Inter Conf on Machine Learning*, 2015.

[14] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc Inter Conf on Learning Representations*, 2014.

[1] http://scikit-learn.org