Multi-directional Recurrent Neural Networks: A Novel Method for Estimating Missing Data

Jinsung Yoon [1] **William R. Zame** [2] **Mihaela van der Schaar** [1 3]

Abstract

Most time-series datasets with multiple data streams have (many) missing measurements that need to be estimated. Most existing methods address this estimation problem either by interpolating within data streams or imputing across data streams; we develop a novel approach that does *both*. Our approach is based on a deep learning architecture that we call a *Multi-directional Recurrent Neural Network* (M-RNN). An M-RNN differs from a bi-directional RNN in that it operates across streams in addition to within streams, and because the timing of inputs into the hidden layers is both lagged and advanced. To demonstrate the power of our approach we apply it to a familiar real-world medical dataset and demonstrate significantly improved performance.

1. Introduction

Missing data (measurements) presents a ubiquitous and challenging problem. Missing measurements are especially common in patient medical data which often incorporates many streams that are sampled at different and irregular times. Because missing medical measurements obscure information crucial to diagnosis, prognosis and treatment, estimating missing medical measurements is an especially important and challenging problem. This paper proposes a novel and extremely effective method of estimating missing measurements that are missing at random. Our method exploits the correlations *within* data streams and the correlation *across* data streams. (Our method is quite general; we use medical data as the example because missing data and correlations across streams and between streams are especially common in medical data.) Our method is based on a novel neural network architecture that we call a Multi-directional Recurrent Neural Network (M-RNN). Our M-RNN executes both interpolation (intra-stream) and imputation (inter-stream) to infer missing data. Like a bi-directional RNN (Bi-RNN) (Graves & Schmidhuber, 2005), an M-RNN operates forward and backward in each data stream – in the *intra-stream directions*. Unlike a Bi-RNN, an M-RNN also operates *across* streams – in the *inter-stream directions*. And, also unlike a Bi-RNN, the timing of inputs into the hidden layers of an M-RNN is lagged in the forward direction and advanced in the backward direction. (To the best of our knowledge, our architecture is the first that operates in this way). To demonstrate the power of our method, we apply it a well-known public real-world medical dataset. We show that our method yields large and statistically significant improvements over previous methods, including familiar interpolation methods such as (Kreindler & Lumsden, 2012; Mondal & Percival, 2010), imputation methods such as (García-Laencina et al., 2010; White et al., 2011; Rehfeld et al., 2011) and RNN-based imputation methods such as (Choi et al., 2015; Lipton et al., 2016; Che et al., 2016) and matrix completion methods such as (Yu et al., 2016; Schnabel et al., 2016).

1.1. Related Works

There are two standard methods to deal with missing at random information in time-series data streams: interpolation and imputation. ((Alaa et al., 2017) proposed a framework to deal with missing not at random data.) Interpolation methods (Kreindler & Lumsden, 2012; Mondal & Percival, 2010) attempt to reconstruct missing data by capturing the temporal relationship *within* each data stream but not the relationships *across* streams. Imputation methods (García-Laencina et al., 2010; White et al., 2011; Rehfeld et al., 2011) attempt to reconstruct missing data by capturing the synchronous relationships *across* data streams but not the

[1]Department of Electrical Engineering, UCLA, USA. [2]Departments of Economics and Mathematics, UCLA, USA. [3]Man Institute, University of Oxford, Oxford, United Kingdom. . Correspondence to: Jinsung Yoon <jsyoon0823@g.ucla.edu>.

ICML 2017 Time Series Workshop, Sydney, Australia, 2017.

temporal relationships *within* streams. Matrix completion methods (Yu et al., 2016; Schnabel et al., 2016) do attempt to use information within and across streams, but (Yu et al., 2016) assumes completely synchronized data so is not applicable in our setting. (Schnabel et al., 2016) is designed for static data and constrains the model as a linear model; thus, it cannot capture the non-linear and time-series characteristics.

RNN's have been used successfully for prediction on the basis of time-series data with missing data and irregular sampling. (Gingras & Bengio, 1996) first replaces all the missing information with a mean value and then uses the feedback loop from the hidden states to update the imputed value while learning the classification problem using a standard RNN for prediction. (Tresp & Briegel, 1998) use the Expectation-Maximization (EM) algorithm to impute the missing values and uses the reconstructed data streams as inputs to a standard RNN for prediction. As with standard imputation methods, the imputation depends only on the synchronous relationships across data streams and not on the temporal relationships within streams. (Parveen & Green, 2002) use a linear model to estimate missing values from the latest measurement and the hidden state of each stream. As with standard interpolation methods, the estimate depends only on the temporal relationships within each stream and not on the relationships across streams.

More recent work addresses both missing values and irregularly sampled time-series data streams (Choi et al., 2015; Lipton et al., 2016; Che et al., 2016; Kim et al., 2017). These papers use the sampling times to capture the informative missingness and time interval information to deal with irregular sampling. They do this by concatenating the measurements, sampling information and time intervals and using the concatenation as the input of an RNN. These papers differ in the replacements they use for missing values. (Choi et al., 2015; Lipton et al., 2016; Kim et al., 2017) replace the missing values with 0, mean values or latest measurements – all of which are independent of either the intra-stream or inter-stream relationships or both. (Che et al., 2016) imputes the missing values using only the most recent measurements, the mean value of each stream, and the time interval. It is not bi-directional.

2. Data Streams

The dataset consists of N arrays of data. It is convenient to use medical language to speak of array n as the information of patient n, so that there are N patients in the training set. For each patient n, we have a multivariate time-series data stream of length T (the length T and the other components may depend on the patient n but for the moment we suppress the dependence on n) that consists of two components: time stamps $\mathcal{S}$ and measurements $\mathcal{X}$. (In many contexts, the data would also include labels/outcomes. However, because we focus here on reconstructing missing data, we treat a label as simply a measurement.)

The *time stamp* $s_t \in \mathbb{R}$ represents the actual time at which the measurements x_t were taken. For convenience we normalize so that $s_1 = 0$; we assume actual times are strictly increasing: $s_{t+1} > s_t$ for $t \leq T-1$. For an irregularly sampled dataset, the difference $s_{t+1} - s_t$ between successive time stamps is not constant.

There are D streams of measurements; each measurement is a real number, but not all measurements may be observed at each time stamp. Hence we view the set of possible measurements at time stamp t as $\mathbb{R}_* = \mathbb{R} \cup \{*\}$; $x_t^d = *$ means that the stream d was not measured at time stamp t; otherwise $x_t^d \in \mathbb{R}$ is the measurement of stream d at time stamp t. $\mathcal{X}$ is the array of measurements of all streams at all time stamps for the patient under consideration.

We define an index m_t^d for missing data; $m_t^d = 0$ if $x_t^d = *$ (not measured) and $m_t^d = 1$ if $x_t^d \in \mathbb{R}$ (measured). For each time stamp t, we write δ_t^d for the actual amount of time that has elapsed since the stream d was measured last; δ_t^d is defined recursively as follows:

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

where $\delta_1^d = 0$. Write $\boldsymbol{\delta}_t$ for the vector of elapsed times at time stamp t and $\Delta = \{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, ..., \boldsymbol{\delta}_T\}$.

The information available for patient n is therefore the pair $(\mathcal{X}_n, \mathcal{S}_n)$ The entire training set therefore is the sets of pairs $\mathcal{D} = \{(\mathcal{X}_n, \mathcal{S}_n)\}_{n=1}^N$. To avoid confusing multiple sub/super-scripts, we use functional notation to identify information about a particular patient, so $x_t^d(n)$ is the measurement of stream d at time t for patient n, etc.

3. M-RNN

Suppose that stream d was not measured at time t, so that that $x_t^d = *$; we want to form an estimate $\hat{x}_t^d$ of what the actual measurement would have been. There are two familiar approaches to this problem: interpolation and imputation. *Interpolation* uses only the measurements $x_{t'}^d$ of the fixed data stream d for other time stamps t' (perhaps both be-

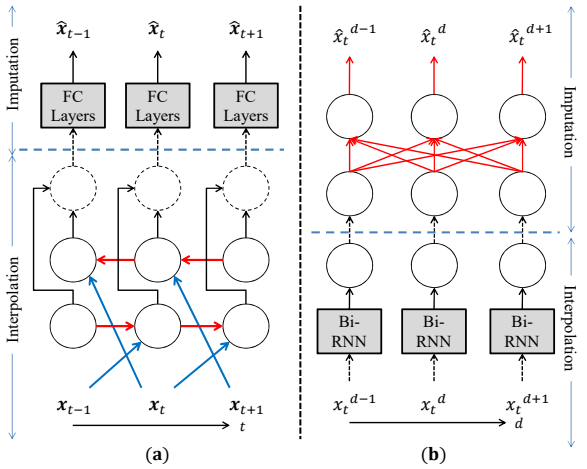

Figure 1. Cross sections of M-RNN architecture. (a) Time-dimension, (b) Feature-dimension

fore and after t) – but ignores the information contained in other data streams d'. *Imputation* uses only the measurements $x_t^{d'}$ at the fixed time t for other data streams d' – but ignores the information contained in other data streams d' at other times $t' \neq t$. Because information is often correlated within data streams *and* across data streams, each of these approaches throws away potentially useful information. Our approach is to form the estimate $\hat{x}_t^d$ using *both* the measurements within the given data stream and the measurements across other data streams. In principle, we could try to form the estimate $\hat{x}_t^d$ by using *all* the information in $\mathcal{D}$. However, this would mean learning a vast number of parameters and hence would require a vast number of training instances, and so would be impractical. Moreover, because of the many parameters, there would be serious danger of over-fitting. Instead, we propose an efficient hierarchical learning framework using a novel RNN architecture that effectively allows us to capture the correlations both within streams and across streams.

3.1. Algorithm/Architecture

An M-RNN consists of 2 blocks: an Interpolation block and an Imputation block; see Fig. 1. Our construction puts the imputation block after the interpolation block in order to use the outputs of the interpolation block to improve the accuracy of the imputation block (multiple inputs of the imputation blocks); the other order would not be helpful.

Error/Loss: The objective of the interpolation and imputation blocks is to minimize the error that would be made in estimating missing measurements. Evidently, we cannot estimate the error of a measurement that is truly missing in the dataset. Instead we fix a measurement that was actually made, remove that measurement, form an estimate for the measurement, and then compute the error between the estimate and the actual measurement (that was deleted) using only the data set $\mathcal{D} - x_t^d$ (i.e. the data set with x_t^d removed). If x_t^d is an actual measurement and $\hat{x}_t^d$ is the estimate formed when x_t^d is removed then the loss is defined as squared error $(\hat{x}_t^d - x_t^d)^2$. The total loss/error for the entire dataset $\mathcal{D}$ is the *mean squared error* (MSE):

$$\mathcal{L}(\{\hat{x}_t^d, x_t^d\}) = \sum_{n=1}^{N} \left[\frac{\sum_{t=1}^{T_n} \sum_{d=1}^{D} m_t^d(n) \times (\hat{x}_t^d(n) - x_t^d(n))^2}{\sum_{t=1}^{T_n} \sum_{d=1}^{D} m_t^d(n)} \right]$$

In our simulations we use *root mean squared error* (RMSE) – i.e. the square root of MSE – to compare performance of M-RNN with various benchmarks.

Interpolation: The objective of the interpolation block is to construct an interpolation function $\mathbf{\Phi}$ that operates *within* a stream. To emphasize that the estimate for x_t^d depends on the data with x_t^d removed, we write $\tilde{x}_t^d = \mathbf{\Phi}(\mathcal{D} - x_t^d)$; but also keep in mind that we are actually only using the data from stream d, not the data from other streams. We construct the estimation function $\mathbf{\Phi}$ using a bi-directional recurrent neural network (Bi-RNN) with a Gated Recurrent Unit (GRU). However, unlike a conventional Bi-RNN (Graves & Schmidhuber, 2005), the timing of inputs into the hidden layer is lagged in the forward direction and advanced in the backward direction: at time t, inputs of forward hidden states come from $t-1$ and inputs of backward hidden states come from $t+1$. Mathematically, we have:

$$\begin{aligned}
\mathbf{o}_t &= \overrightarrow{W}\overrightarrow{\mathbf{h}}_t + \overleftarrow{W}\overleftarrow{\mathbf{h}}_t + \mathbf{c}_o \\
\overrightarrow{\mathbf{h}}_t &= (1 - \overrightarrow{\mathbf{z}}_t) \odot \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{z}}_t \odot \overrightarrow{\hat{\mathbf{h}}}_t \\
\overleftarrow{\mathbf{h}}_t &= (1 - \overleftarrow{\mathbf{z}}_t) \odot \overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{\mathbf{z}}_t \odot \overleftarrow{\hat{\mathbf{h}}}_t \\
\overrightarrow{\mathbf{z}}_t &= \sigma(\overrightarrow{W}_z\mathbf{x}_{t-1} + \overrightarrow{U}_z\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{V}_z\boldsymbol{\delta}_{t-1} + \overrightarrow{\mathbf{c}}_z) \\
\overleftarrow{\mathbf{z}}_t &= \sigma(\overleftarrow{W}_z\mathbf{x}_{t+1} + \overleftarrow{U}_z\overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{V}_z\boldsymbol{\delta}_{t+1} + \overleftarrow{\mathbf{c}}_z) \\
\overrightarrow{\hat{\mathbf{h}}}_t &= \phi(\overrightarrow{W}_h\mathbf{x}_{t-1} + \overrightarrow{U}_h(\overrightarrow{\mathbf{r}}_t \odot \overrightarrow{\mathbf{h}}_{t-1}) + \overrightarrow{V}_h\boldsymbol{\delta}_{t-1} + \overrightarrow{\mathbf{c}}_h) \\
\overleftarrow{\hat{\mathbf{h}}}_t &= \phi(\overleftarrow{W}_h\mathbf{x}_{t+1} + \overleftarrow{U}_h(\overleftarrow{\mathbf{r}}_t \odot \overleftarrow{\mathbf{h}}_{t+1}) + \overleftarrow{V}_h\boldsymbol{\delta}_{t+1} + \overleftarrow{\mathbf{c}}_h) \\
\overrightarrow{\mathbf{r}}_t &= \sigma(\overrightarrow{W}_r\mathbf{x}_{t-1} + \overrightarrow{U}_r\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{V}_r\boldsymbol{\delta}_{t-1} + \overrightarrow{\mathbf{c}}_r) \\
\overleftarrow{\mathbf{r}}_t &= \sigma(\overleftarrow{W}_r\mathbf{x}_{t+1} + \overleftarrow{U}_r\overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{V}_r\boldsymbol{\delta}_{t+1} + \overleftarrow{\mathbf{c}}_r)
\end{aligned}$$

where $\odot$ is element-wise multiplication, σ is the sigmoid function, ϕ is $\tanh$ function, and arrows indicate direction.

Table 1. Performance comparison for missing value estimation

RMSE: Mean ± Std Dev (% Gain of M-RNN)	
M-RNN	**0.0137 ± 0.0013 (-)**
(Choi et al., 2015)	0.0337 ± 0.0012 (59.3%)
(Lipton et al., 2016)	0.0295 ± 0.0009 (53.6%)
(Che et al., 2016)	0.0292 ± 0.0013 (53.1%)
Spline Interpolation	0.0735 ± 0.0012 (81.4%)
Cubic Interpolation	0.0279 ± 0.0013 (50.9%)
MICE Imputation	0.0611 ± 0.0011 (77.6%)
Kernel Imputation	0.0556 ± 0.0011 (75.4%)
EM Imputation	0.0467 ± 0.0014 (70.7%)
Matrix Completion	0.0311 ± 0.0013 (55.9%)

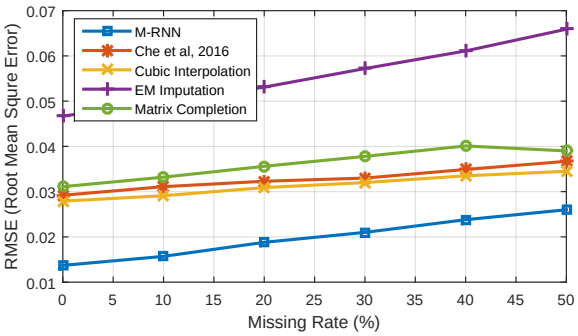

Figure 2. RMSE for M-RNN and Best Benchmarks

The output $\mathbf{o}_t$ is the interpolated value $\tilde{\mathbf{x}}_t$. In this interpolation block, we are only using/capturing the temporal correlation within each stream. As a consequence, the matrices U, V, W are diagonal. Hence the total number of parameters that must be learned is linear in the number of streams D. See the Interpolation component of Fig. 1.

Imputation: The objective of the imputation block is to construct an imputation function $\mathbf{\Psi}$ that operates *across* streams. Again, we write $\check{x}_t^d = \mathbf{\Psi}(\mathcal{D} - x_t^d)$, but keep in mind that now we are using only data at time stamp t, not data from other time stamps. We construct the function $\mathbf{\Psi}$ to be independent of the time stamp t; so we construct it using fully connected layers (FC); see the Imputation component of Fig 1:

$$\mathbf{o}_t = W\mathbf{h}_t + \mathbf{c}_o \qquad \mathbf{h}_t = U\mathbf{x}_t + V\tilde{\mathbf{x}}_t + Q\mathbf{m}_t + \mathbf{c}_h$$

where $\mathbf{o}_t = \hat{\mathbf{x}}_t$ and the diagonal entries of U are zero because we do not use x_t^d to estimate $\hat{x}_t^d$. We use multiple deeply stacked FC layers. In principle, any activation functions (ReLu, tanh, linear) could be used. In experiments, we found that linear activation function worked best.

We jointly learn the functions $\mathbf{\Phi}$ and $\mathbf{\Psi}$ using the stacked networks of Bi-RNN and FC layers.

$$\mathbf{\Phi}^*, \mathbf{\Psi}^* =$$
$$\arg\min_{\mathbf{\Phi},\mathbf{\Psi}} \mathcal{L}(\{\mathbf{\Psi}\Big(\{x_t^d, \mathbf{\Phi}\Big(\{x_t^d, m_t^d, \delta_t^d\}_{t=1}^T\Big), m_t^d\}_{d=1}^D\Big), x_t^d\})$$

We refer to the entire structure as a *Multi-directional Recurrent Neural Network (M-RNN)*; see Fig.1.

4. Experiments

In this section, we evaluate the performance of M-RNN using Medical Information Mart for Intensive Care (MIMIC)-III (Johnson et al., 2016), a public real-world dataset of electronic health records with values missing at random. Our experimental results present the performance gain of our M-RNN algorithm in estimating missing valuesm in comparison to benchmarks. (We describe all configurations of the our M-RNN and the other algorithms in the online appendix.)

Simulation To evaluate the performance of the M-RNN algorithm in estimating missing values, we compare with familiar interpolation methods (Spline and Cubic) and imputation methods (MICE, Kernel, and EM) and state-of-the-art RNN-based imputation methods (Choi et al., 2015; Lipton et al., 2016; Che et al., 2016) methods, and matrix completion method (Schnabel et al., 2016). We use root mean square error (RMSE) as the performance metric.

We conduct a number of experimental comparisons. The basic comparison uses the entire MIMIC-III dataset. Table 1 shows the mean and standard deviation of RSME for our method and benchmarks and the percentage improvement of RMSE for M-RNN over the benchmarks. Notice that the RMSE for M-RNN is less than half that of the best benchmark. All the improvements are statistically significant (p-value < 0.05).

To evaluate the performance of our method in comparison with benchmarks when there is more missing data (which makes the problem of estimating missing measurements more difficult) we construct sub-samples of the MIMIC-III data set by randomly removing 10%, 20%, 30%, 40%, 50% of the actual data and carrying out the same exercise on the smaller dataset that remains. The graphs in Fig. 2 show the performance of M-RNN against the *best* benchmarks in these settings; as can be seen M-RNN continues to substantially out-perform the benchmarks. Note that as the amount of missing data increases the improvement of M-RNN over the imputation benchmark(s) increases but the improvement over the interpolation benchmarks decreases.

References

Alaa, Ahmed M, Hu, Scott, and van der Schaar, Mihaela. Learning from clinical judgments: Semi-markov-modulated marked hawkes processes for risk prognosis. In *ICML*, 2017.

Che, Zhengping, Purushotham, Sanjay, Cho, Kyunghyun, Sontag, David, and Liu, Yan. Recurrent neural networks for multivariate time series with missing values. *arXiv preprint arXiv:1606.01865*, 2016.

Choi, Edward, Bahadori, Mohammad Taha, and Sun, Jimeng. Doctor ai: Predicting clinical events via recurrent neural networks. *arXiv preprint arXiv:1511.05942*, 2015.

García-Laencina, Pedro J, Sancho-Gómez, José-Luis, and Figueiras-Vidal, Aníbal R. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2):263–282, 2010.

Gingras, Francois and Bengio, Y. Recurrent neural networks for missing or asynchronous data. In *Proc NIPS*, volume 8, 1996.

Graves, Alex and Schmidhuber, Jürgen. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5): 602–610, 2005.

Johnson, Alistair EW, Pollard, Tom J, Shen, Lu, Lehman, Li-wei H, Feng, Mengling, Ghassemi, Mohammad, Moody, Benjamin, Szolovits, Peter, Celi, Leo Anthony, and Mark, Roger G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

Kim, Han-Gyu, Jang, Gil-Jin, Choi, Ho-Jin, Kim, Minho, Kim, Young-Won, and Choi, Jaehun. Recurrent neural networks with missing information imputation for medical examination data prediction. In *Big Data and Smart Computing (BigComp), 2017 IEEE International Conference on*, pp. 317–323. IEEE, 2017.

Kreindler, David M and Lumsden, Charles J. The effects of the irregular sample and missing data in time series analysis. *Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data*, pp. 135, 2012.

Lipton, Zachary C, Kale, David C, and Wetzel, Randall. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. *arXiv preprint arXiv:1606.04130*, 2016.

Mondal, Debashis and Percival, Donald B. Wavelet variance analysis for gappy time series. *Annals of the Institute of Statistical Mathematics*, 62(5):943–966, 2010.

Parveen, Shahla and Green, Phil. Speech recognition with missing data using recurrent neural nets. In *Advances in Neural Information Processing Systems*, pp. 1189–1195, 2002.

Rehfeld, Kira, Marwan, Norbert, Heitzig, Jobst, and Kurths, Jürgen. Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*, 18(3):389–404, 2011.

Schnabel, Tobias, Swaminatan, Adith, Singh, Ashudeep, Chandak, Navin, and Joachims, Thorsten. Recommendations as treatments: debiasing learning and evolution. *Proc ICML*, 2016.

Tresp, Volker and Briegel, Thomas. A solution for missing data in recurrent neural networks with an application to blood glucose prediction. *Advances in Neural Information Processing Systems*, pp. 971–977, 1998.

White, Ian R, Royston, Patrick, and Wood, Angela M. Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine*, 30(4):377–399, 2011.

Yu, Hsiang-Fu, Rao, Hikhil, and Dhillon, Inderjit S. Temporal regularized matrix factorization for high-dimensional time series prediction. *Proc NIPS*, 2016.