The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

Recover Missing Sensor Data with Iterative Imputing Network

Jingguang Zhou, Zili Huang
Shanghai Jiao Tong University
{wintersky, huangziliandy}@sjtu.edu.cn

Abstract

Sensor data has been playing an important role in machine learning tasks, complementary to the human-annotated data that is usually rather costly. However, due to systematic or accidental mis-operations, sensor data comes very often with a variety of missing values, resulting in considerable difficulties in the follow-up analysis and visualization. Previous work imputes the missing values by interpolating in the observational feature space, without consulting any latent (hidden) dynamics. In contrast, our model captures the latent complex temporal dynamics by summarizing each observation's context with a novel Iterative Imputing Network, thus significantly outperforms previous work on the benchmark Beijing air quality and meteorological dataset. Our model also yields consistent superiority over other methods in cases of different missing rates.

Introduction

Big Data is indispensable for the development of machine learning (Manyika et al. 2011). Besides human-annotated data, geo-distributed sensors are great source for data collection, which benefits the development of machine learning methods in understanding the environmental dynamics. In a common sensing or crowd sensing campaign (Chong and Kumar 2003), sensors at different locations collect the environmental data during a time period. However, most sensing campaigns suffer from systematic or accidental missing data mechanism, like broken sensors, communication errors and etc. Such unfortunate information loss throws importance upon imputing missing values in the sensor data.

Sensor data recovery is a great challenge due to the remarkable portion of missing entries and their stochastic distribution. A handful of studies attempt to leverage the locality in the observational feature space via conventional methods like inverse distance weighting (Chen and Liu 2012) or ARMA (Valipour, Banihabib, and Behbahani 2013) or nearest neighbors (Pan and Li 2010). Such methods yield unsatisfactory results as they fail to capture the latent, complex, and potentially higher-order temporal dynamics. In contrast, we aim to capture such dynamics in the latent (hidden) feature space by neural networks, which have proven rather effective in learning latent dynamics of time-series data (Längkvist, Karlsson, and Loutfi 2014; Mei and Eisner 2017).

However, common deep learning procedure cannot directly be used with incomplete training data. We develop a flexible scheme to deal with this—-first initialize the entries using simple statistic estimates, and then update the estimated value via a novel multi-layer Iterative Imputing Network (IIN). The core component of our Iterative Imputing Network is a multi-layer Long Short-Term Memory (LSTM) network, which consumes a sequence of time-stamped items and summarizes the representation and context information for each of them. An output layer is then stacked on this LSTM and projects the representation of any missing observation to a readable imputation. We propose to use two different versions of LSTM—the standard LSTM (Hochreiter and Schmidhuber 1997) for regularly sampled sensor data and Phase-LSTM (Neil, Pfeiffer, and Liu 2016) for the irregularly sampled case. We call it Imputing Network (IN).

Our novel Iterative Imputing Network (IIN) is a multi-level cascade of Imputing Networks (IN) that share the same set of weights, with the output imputation of any member IN block being fed into the higher-level one as input. It is thus mathematically equivalent to iteratively training the same Imputing Network with the same sequence, until the imputation accuracy achieves a satisfactory level.

Why is this important? Because by iteratively connecting (training) the network, the issue of data sparsity, known as a natural enemy of neural models, is well handled—the Imputation Network is able to gradually adapt itself by iteratively refining its missing value imputation on one single sequence sample. Note that such data sparsity issue is especially troublesome in the task of missing data imputation, because 1) missing values naturally and effectively reduce data adequacy; 2) missing values sometimes are clustered together forming missing blocks, which are even more challenging to deal with.

Our Iterative Imputing Network has essential advantages over previous methods. First, our model summarizes higher-order temporal dynamics in the time-series, by representing each time-stamped observation with a deep neural network, while previous methods highly reply on locality in the observational feature space and could only summarize low-order (if more than one) temporal dependency in the time-series. Second, by representing the sequences in hidden space and

iteratively refining the missing value imputation, our model better deals with missing blocks. Our model is consistent with the imputations within a missing block, and effectively adopt more information than the previous methods that skip missing values. According to these advantages, our model outperforms all previous methods on a hard benchmark Beijing air quality and meteorological dataset. Moreover, we demonstrate with our experiments that the superiority of our model is consistent with varying missing data rates.

In summary, our main contributions are as follows:

- We propose a practical scheme for sensor data recovery, enabling the use of deep learning procedure by initializing missing entries via flexible methods.
- We design a novel Iterative Imputing Network (IIN), capturing the high-order latent temporal dynamics and iteratively refining the estimation of missing values.
- Our method significantly improves the state-of-the-art result on a hard benchmark, and shows robustness with varying missing rates.

Related Work

In the following, we review existing works related to our problem, including: (1) sensor data recovery; (2) deep learning for time series.

Sensing Data Recovery

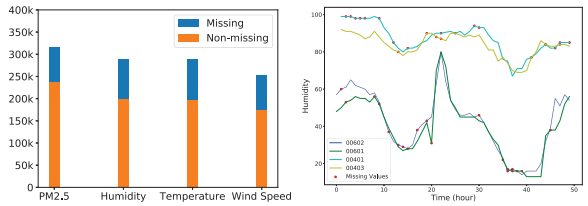
When wireless sensor network emerged, (Doherty and others 2000) pointed out the importance of data recovery. Once all missing values have been imputed, the dataset can then be analyzed using standard techniques for complete data. (Yozgatligil et al. 2013; Lee, Kulic, and Nakamura 2008) considered the temporal dependencies of sensing data series using statistical analysis like ARMA. Some studies include the spatial cue into the missing data recovery. (Pan and Li 2010) presented a K-nearest neighbor method for jointly spatial and temporal data imputation. (Yi et al. 2016) considered spatial similarities together with their temporal similarities. Nevertheless, both of them only captured features on the surface, falling short in learning the internal dynamic of the temporal data. (Gruenwald et al. 2010) applied tree-based data mining techniques to handle missing data on real-life and synthetic datasets. (Lindström et al. 2014b) utilized matrix analysis, while (Sorjamaa et al. 2010) proposed a linear projection method called empirical orthogonal functions. However, their methods did not show enough robustness to the high rates and random distribution of missing data. Additionally, our approaches do not explicitly model the spatial similarities, since they can involve great uncertainties or noise. Instead, we feed all sensors' data into one network without seperating each sensor, enabling our network to benefit from shared trends and common property of different sensors.

Deep Learning for Series

Deep learning has been known for its great capability of learning data representations automatically, instead of using hand-craft features. Recurrent neural networks (RNN) and Long short-term memory (LSTM) networks (Gers, Eck, and Schmidhuber 2000; Malhotra et al. 2015) showed the efficacy for time series regression. However, few studies dive deep in dealing with missing values. (Parveen and Green 2004) used binary indicators to handle missing values representing a pause between two sentences or a possibly interrupt for speech signals. Their missing values did not contain much information, which could cause much degradation of accuracy. They did not truly solve missing data recovery, but just ingore that. In contrast, our scheme aims to tackle the problem of missing data recovery, which has great significance due to the great amount but low quality of sensor data. Inspired by (Dai, He, and Sun 2016) which proposed a segmentation cascade model in computer vision, we design our multi-level cascade Iterative Imputing Network (IIN) to model the recovery of time series.

Overview

Sensor data usually suffer from missing values because of errors in data collection and transmission. The missing entries have large and random distribution, illustrated by Figure 1a. It is hard for us to utilize the sensor data for subsequent analysis and visualization unless we tackle the prevalent innate missing entries in the dataset.



(a) (b)

Figure 1: (a) The proportion of missing data in four training datasets. (b) Humidity data collected by four sensors in two different locations. Red circle represents missing entries.

To formulate the problem, we treat the series of data collected by one of the sensors as a sequence $\mathcal{X}$. Partial entries $\mathcal{X}_m = \{x_{m1}, x_{m2}, ..., x_{mk}\}$ are missing, while the other entries $\mathcal{X}_r = \mathcal{X}/\mathcal{X}_m$ are numerical values like temperatures and humidities. $\mathcal{X}_m$ are in an unknown random distribution and some of them locate in consecutive values, forming a block missing. Our goal is to learn from $\mathcal{X}/\mathcal{X}_m$ and get a better-estimated value for missing readings $\mathcal{X}_m$. Based on $\mathcal{X}$, we split the data into the training set $\mathcal{X}^{tr}$ and test set $\mathcal{X}^{te}$. For testing set $\mathcal{X}^{te}$, let $\mathcal{X}_r^{te}$ denote $\mathcal{X}^{te}/\mathcal{X}_m^{te}$. Then we extract a portion of non-missing entries denoted by $\mathcal{X}_{testGT}$ from $\mathcal{X}_r^{te}$ as ground truth, and learn how to recover from $\mathcal{X}_r^{te}/\mathcal{X}_{testGT}$. Our intuition is that first initialize the entries using simple methods with few costs, and then apply deep learning model to learn and give prediction based on high-order latent temporal dynamics. The mean absolute error (MAE) and mean relative error (MRE) are used as metrics evaluating the quality of recovery data.

The Model

Imputing Network

Imputing Network, as the core component of our model, summarizes the context of each missing value by consuming its left and right neighboring observations or imputations with a forward and backward recurrent neural network (RNN) respectively, as shown in Figure 2. An output layer is then stacked upon the representations extracted by these RNNs and learns to impute the current missing value.

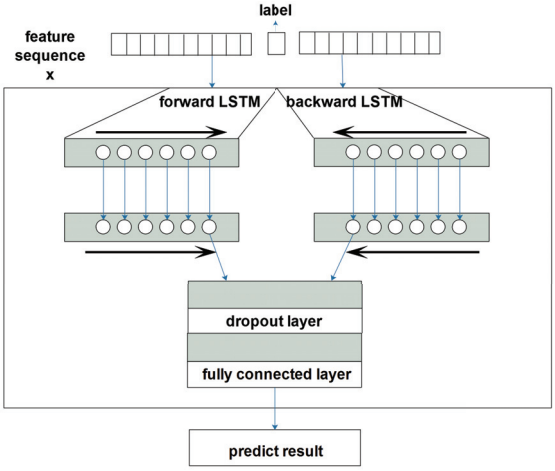


Figure 2: Architecture of Imputing Network.

We choose Long Short-Term Memory (LSTM) networks as our RNNs, because they have proven very effective at sequential modeling—they are able to handle long-range dependency along sequences and to prevent the gradients from exploding or vanishing with the memory cell.

Formally, our multi-layer LSTMs recurrently consume context of each to-be-imputed position t in the sequence, as follow:

$$h_s^f = LSTM^f(h_{s-1}, x_s) \tag{1a}$$

$$h_u^b = LSTM^b(h_{u+1}, x_u) \tag{1b}$$

where the superscripts f and b denote 'forward' and 'backward' respectively, x is the (possibly imputed) observation at each position, and $s \leq t \leq u$. After s and u both reach t, the imputation $\hat{x}_t$ is computed by passing h_t^f and h_t^b through the output layer, as follow:

$$\hat{x}_t = OUTPUT(h_t^f, h_t^b) \tag{2}$$

. In this paper, we adopt two versions of LSTMs, as shown in Figure 3. The standard LSTM (Hochreiter and Schmidhuber 1997) is used to model regularly sampled sensor data. The update equations of standard LSTM is as follows.

$$i_t = \sigma_i(x_t W_{xi} + h_{t-1} W_{hi} + w_{ci} \odot c_{t-1} + b_i) \tag{3a}$$

$$f_t = \sigma_f(x_t W_{xf} + h_{t-1} W_{hf} + w_{cf} \odot c_{t-1} + b_f) \tag{3b}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(x_t W_{xc} + h_{t-1} W_{hc} + b_c) \tag{3c}$$

$$o_t = \sigma_o(x_t W_{xo} + h_{t-1} W_{ho} + w_{co} \odot c_t + b_o) \tag{3d}$$

$$h_t = o_t \odot \sigma_h(c_t) \tag{3e}$$

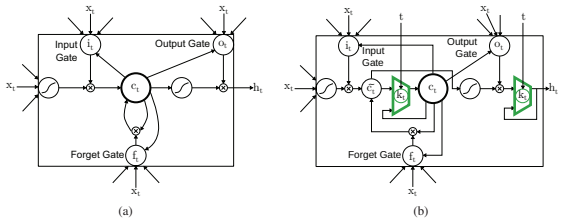


Figure 3: (a) standard LSTM: discrete-time; (b) phased LSTM: continuous-time.

Phase-LSTM (Neil, Pfeiffer, and Liu 2016) incorporates time gate into LSTM cells in order to process irregularly sampled data, which is triggered by events generated in continuous-time [1]. The update equations of Phase-LSTM are shown as follows:

$$\widetilde{c_j} = f_j \odot c_{j-1} + i_j \odot \sigma_c(x_j W_{xc} + h_{j-1} W_{hc} + b_c) \tag{4a}$$

$$c_j = k_j \odot \widetilde{c_j} + (1 - k_j) \odot c_{j-1} \tag{4b}$$

$$\widetilde{h_j} = o_j \odot \sigma_h(\widetilde{c_j}) \tag{4c}$$

$$h_j = k_j \odot \widetilde{h_j} + (1 - k_j) \odot h_{j-1} \tag{4d}$$

Iterative Imputing Network

To learn refined imputation and deal with possible data sparsity caused by missing values, we propose a novel Iterative Imputing Network (IIN), which is a multi-level cascade of Imputing Networks that share the same set of weights, with the output imputation of any member IN block being fed into the higher-level one as input. This is important (as we can show shortly in experiments), because such design enables the Imputing Network to gradually adapt itself by iteratively refining its missing value imputation on one single sequence sample! This benefits the model learning by 1) jointly utilizing the information from not only visible observations but also previously imputed missing values and 2) well handling (potential) data sparsity caused by missing data.

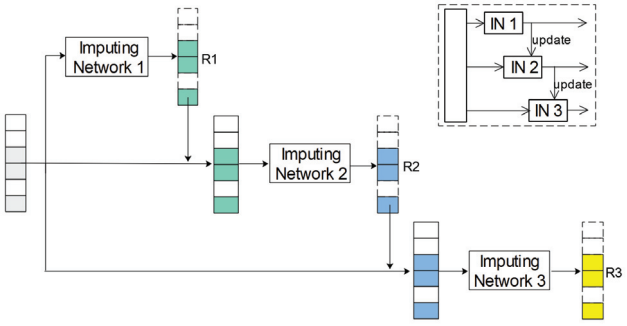


Figure 4: A three-stage cascade. On stage 2, series in which missing entries are updated by Imputing Network 1 are used as input to Imputing Network 2.

[1] For convenience, we call both two options LSTM and only differentiate them when it is needed.

Figure 4 illustrates the INN architecture. Note that this design is mathematically equivalent to iteratively training the same Imputing Network with the same sequence, until the imputation accuracy achieves a satisfactory level. Therefore, we naturally have two options for training the model—training the Iterative Imputing Model as a whole by gradient-based methods or iteratively training the same Imputing Network. As the former option is straightforward, we only elaborate the latter one in this section.

Our iterative recipe to train the model is illustrated in Algorithm 1. This algorithm is essentially analogous to expectation-maximization (EM) algorithm: using currently-estimated model weights to impute the missing values (expectation) and then updating these weights by maximizing the log-likelihood given the observational data (maximization) (Allison 2002).

Algorithm 1: Iterative Training Recipe.

Input: Origin Data Series M; Iterations $iter\_num$
Output: Final Recovery Data Matrix

Initialize series T_0 from M with statistical methods;
Fill the remaining missing entries in T_0 with nearest data records in time domain;
$i \leftarrow 0$;
repeat
  Extract valid feature-label pairs from T_i;
  Train Imputing Network $model_i$;
  Predict the missing values in T_i using $model_i$;
  $T_{i+1} \leftarrow$ update the missing entries in T_i;
  $i \leftarrow i + 1$;
until $i \geq iter\_num$;
return series T_{i+1};

Experimental Analysis

In this section, we evaluate the effectiveness of our model on the benchmark Beijing air quality and meteorological dataset (Yu Zheng 2013), elaborate the experimental details, and analyze the results.

Dataset Preparation

We conduct experiments on the Beijing air quality and meteorological dataset. The geo-distributed air quality data and meteorological data was recorded every hour. We select the subset of PM2.5 from the air quality dataset and select the subsets of temperature (TEMP), humidity (HUM) and wind speed (WS) from the meteorological dataset. Figure 1b shows humidity data series collected by 4 sensors in two different places. Sensor 00601 and 00602 are close to each other, while Sensor 00401 and 00403 are in the adjacent areas. Looking from the time dimension, the sensing data do not show an obvious periodicity. The sensing data may be affected by sparse asynchronous streams of events, which makes the learning and understanding of the internal data pattern a very challenging task. Worse still, there are noticeable missing entries in sensing dataset because of collecting errors and network errors, illustrated by Figure 1a.

Besides the original missing entries in the sensing dataset, we need to prepare our training set and testing set by setting aside some entries as ground-truth. For PM2.5 dataset, we apply the method in (Yi et al. 2016) to generate missing values. First, we record the positions of all missing values in each month's data. Then we manually remove the values on the same position in the next month. (For instance, if the entry for a sensor at 2014-05-04 14:00:00 is missing, then we drop out the value of this sensor at 2014-06-04 14:00:00). For the other three datasets(temperature, humidity and wind speed), we randomly set aside 20% of the total non-missing values as the ground-truth of missing entries. In our experiment, we use the sensor recordings in the 3, 6, 9 and 12 month as testing set and the rest as training set.

Anchor Selection

A common practice in sensor data recovery is to operate within a sliding window that is centered at the to-be-imputed entry. Such a sliding window is usually called the *anchor* (Ren et al. 2015) of this entry [2] [3]. An example anchor of size 7 in the dataset is shown in Figure 5. As we can see, the to-be-imputed entry is neighbored by its left (previous) and right (subsequent) context. The special value NA denotes the missing observations.

	t1	t2	t3	t4	t5	t6	t7	t8	...
s1	138	124	127	129	NA	NA	NA	120	
s2	89	NA	88	100	109	NA	127	114	
s3	NA	121	NA	NA	144	NA	112	108	
...									

Figure 5: Anchor (blue) w.r.t. an entry (gray).

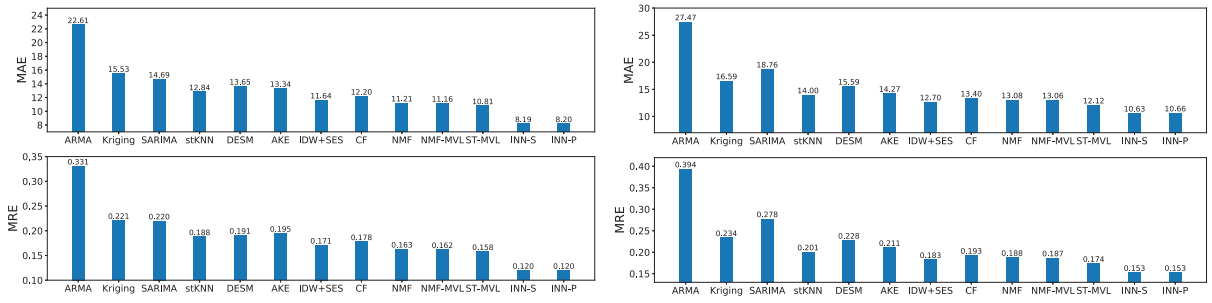
We aim to train our model only with anchors that carry adequate information about the dynamics. Therefore, we define each anchor to be valid for use, only if more than 50% of the entries are observable (i.e. not missing). As with how to fill the missing blanks during training, we use the imputed values estimated by our model (with recently updated weights), which will be shown more effective than other common practices shortly in the results section.

Implementation Details

In our Imputing Network, we use a two-layer forward and backward LSTM to model the latent patterns of series. The first layer takes a sequence with a half of the window size as input, then outputs hidden units each of which is of 50 dimensions. The second layer will output the last 100-dimensional unit. We concatenate the outputs of forward and backward LSTM and then use dropout at a rate of 0.3. We set aside $1/10$ of the data in training set as our validation set

[2] An anchor is associated with a window size, which could be large in order to cover several sensing cycles.

[3] Anchors can be extended to multi-scale, i.e. combining anchors with multiple window sizes, or covering multiple sensors at the same time, but here we discuss the basic anchor for a single series.



(a) (b)

Figure 6: Recovery error of different methods in two scenarios (a) general missing; (b) overall missing.

and maintain the models which show best performance on the validation set.

To train the Imputation Network, we use mean absolute error (MAE) as our loss function and we apply Nesterov-accelerated adaptive moment estimation (Nadam) algorithm (Sutskever et al. 2013; Dozat 2016) to optimize our neural network. Nadam incorporates Nesterov Momentum into Adam, making it consistently outperform Adam and RMSProp. For the LSTM cells, we initialize kernel weight matrix in a glorot uniform distribution, the recurrent kernel weight matrix in an orthogonal distribution, and set the bias to be zeros.

To implement IIN, a multi-level cascade of Imputing Networks, we maintain the index of the missing entries. We iteratively update the missing entries and pass the output into Imputing Network for two or three times. To update the missing entries, we maintain their indices. Cascade IIN will refine the data recovery as the number of iteration increases.

The entire process can also be seen as the adaption of EM iterative strategy to deep learning. At the first round, the large ratio of missing reading in a sensing dataset makes it unable to apply deep learning methods. Therefore, we actually use some combinations of statistical values of the data, which can be viewed as an unsupervised feature extraction. This is similar to E step. After we impute the missing entries from observed data, we get "artificially" intact data and pass them to the LSTM-based IIN networks. The Imputing Network will learn the dynamics of the sensor data and give the most probable outputs for missing entries. This is similar to M step. After the first round, we could omit the E step—-use the network outputs in the previous round as input data and do M step directly.

Recovery Accuracy and Error

We measure the performance by Mean Absolute Error (MAE) and Mean Relative Error (MRE),

$$MAE = \frac{\sum_{i=1}^{N} |x_{t_i} - \hat{x}_{t_i}|}{N}, MRE = \frac{\sum_{i=1}^{N} |x_{t_i} - \hat{x}_{t_i}|}{\sum_{i=1}^{N} x_{t_i}}$$

where $\hat{x}_{t_i}$ and x_{t_i} is the estimated and ground-truth value respectively at time t_i (with index i), and N is the total number of observations.

Missing values occur in sensing dataset in a stochastical way. To eliminate the dominance of extreme cases and corner cases on MAE and MRE, we compute the mean error in a general scenario, which we denote as general missing. Similar to (Yi et al. 2016), for the general missing we do not consider the spatial missing block (the missing values that records of all sensors are simultaneously absent) and the temporal missing block (the records of a sensor are missing in a certain length of time window, 11 in our experiments). For fair comparisons, we compute MAE and MRE for all missing entries, which scenario we denote as overall missing.

We compare our method with 10 baselines, including ARMA (Valipour, Banihabib, and Behbahani 2013), stKNN (Pan and Li 2010), Kriging (Wu and Li 2013), SARIMA (Yozgatligil et al. 2013), DESM (Gruenwald et al. 2010), AKE (Pan and Li 2010), IDW+SES (Gardner 1985), CF (Sarwar et al. 2001; Su and Khoshgoftaar 2009), NMF (Lee and Seung 2001; Lindstrom et al. 2014a), ST-MVL (Yi et al. 2016). Prior best results on Beijing air quality and meteorological dataset are achieved by ST-MVL, which use statistical methods to extract four features from the observed feature space.

We evaluate two types of IIN, based on standard LSTM and phased LSTM respectively. We denote them as IIN-S and IIN-P. Table 6 shows the comparison of different missing scenarios between different methods [4]. As we can see, IIN outperform other baselines significantly in two missing scenarios. ST-MVL, the prior best work, improves 0.3 on general missing than before. In contrast, IIN's MAE improves 2.6 on the results of ST-MVL. Moreover, IIN predict estimation for all spatial blocks and temporal blocks, while prior methods skip some extremely hard missing entries when computing the accuracy. In spite of this, IIN still outperforms significantly the prior methods.

Performance with Different Missing Rates

We consider the impact of different missing rates over our training process. The miss rate refers to the ratio of the number of missing entries over the total entries. Higher missing rate bring severe bias to data recovery. Here we prepare our

[4] Results of different methods come from (Yi et al. 2016).

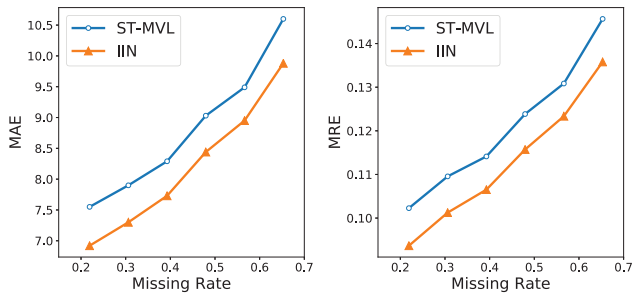


Figure 7: Impact of missing rates over the recovery error.

dataset by randomly dropping out the data records with different missing rates. Illustrated by Figure 7, Our methods are always better than ST-MVL.

Missing Value Initialization Analysis

To kick off the training of our model, we fill the missing entries with appropriate initialization. Our schemes are flexible in different initialization methods, and then refine the estimation using deep learning approaches. We initialize the missing entries using the method proposed by (Yi et al. 2016), which regresses each missing entry initialization on four commonly used corpus-level statistics.

In Figure 8, we also evaluate the impacts of other initialization methods on our recovery accuracy. We compute the recovery error after passing the initialized data into Imputing Network. ST-MVL is the best initialization method, probably because it combines four statistical factors. Whatever initialization methods are used, IIN can always help refine the estimation of the results. (For MAE, AKE: 14.27 to 11.77; CF: 13.40 to 11.41; ST-MVL: 12.12 to 10.66.) We should note that even if we initialize with AKE, IIN can further help to reduces the MAE to 11.77, lower than ST-MVL 12.12 without IIN.

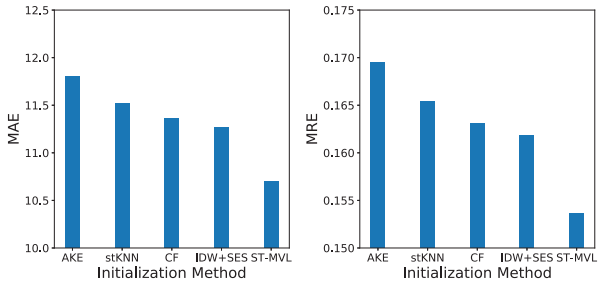


Figure 8: Impact of initialization method based on PM2.5.

Separate or Mix Different Sensors?

In our recovery scheme, we utilize data from all sensors in order for jointly training of our neural networks. However, it seems natural to separately train and predict for the data of different sensors, in order to avoid their data patterns interfering with each other. We experimented in this way, applying our schemes on separate sensors and taking an average of their error. We denote this separate version as IIN(sep). The results are illustrated as Table 1. IIN shows better performance on all datasets than the state-of-art methods ST-MVL. Nevertheless, IIN(sep) is not always better than ST-MVL. IIN(sep) shows an edge over ST-MVL on PM2.5 dataset, but fails on temperature dataset.

Table 1: Results on different datasets.

Dataset		PM2.5	TEMP	HUM	WS
ST-MVL	MAE	12.12	0.68	3.37	1.89
	MRE	0.1740	0.0459	0.0591	0.2985
IIN(Sep)	MAE	10.78	0.74	3.10	1.88
	MRE	0.1558	0.0496	0.0544	0.2958
IIN	MAE	10.63	0.63	2.90	1.87
	MRE	0.1531	0.0422	0.0509	0.2953

Therefore, mixing sensor data does not introduce much noise. Instead we argue that this enables our network to benefit from shared trends or common properties of different sensors. Some geo-distributed sensor data have strong correlations, which benefits the recovery process on the missing data. For example, if we need to impute the missing values from an anchor of sensor 1, we find its neighboring sensor 2 has complete data of its anchor in the same period. Then the anchor of sensor 2 will help if we pass it into IIN, which gives prediction based on common data patterns in the latent feature space.

Conclusion

Besides the human-annotated data that is usually rather costly, sensor data has been for a long time playing an important role in machine learning tasks. However, systematic or accidental mis-operations often result in a variety of missing data, which significantly adds up the noise in the collected dataset. While previous work only imputes the missing values by interpolating in the observational feature space, we aim to model the latent (hidden) temporal dynamics by summarizing each observation's context with a novel Iterative Imputing Network. Our model significantly outperforms previous work on the benchmark Beijing air quality and meteorological dataset, and also yields consistent superiority over other methods in cases of different missing rates.

Acknowledgments

We sincerely thank Hongyuan Mei for many helpful discussions and comments on the manuscript. We also thank Xinsong Zhang and Weijia Jia for their introduction of the dataset and their encouragement.

References

Allison, P. D. 2002. Missing data: Quantitative applications in the social sciences. *British Journal of Mathematical and Statistical Psychology* 55(1):193–196.

Chen, F.-W., and Liu, C.-W. 2012. Estimation of the spatial rainfall distribution using inverse distance weighting (idw) in the middle of taiwan. *Paddy and Water Environment* 10(3):209–222.

Chong, C.-Y., and Kumar, S. P. 2003. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE* 91(8):1247–1256.

Dai, J.; He, K.; and Sun, J. 2016. Instance-aware semantic segmentation via multi-task network cascades. In *Computer Vision and Pattern Recognition (CVPR)*, 3150–3158.

Doherty, L., et al. 2000. Algorithms for position and data recovery in wireless sensor networks. *UC Berkeley EECS Masters Report*.

Dozat, T. 2016. Incorporating nesterov momentum into adam.

Gardner, E. S. 1985. Exponential smoothing: The state of the art. *Journal of Forecasting* 4(1):1–28.

Gers, F. A.; Eck, D.; and Schmidhuber, J. 2000. Applying lstm to time series predictable through time-window approaches. *international conference on artificial neural networks* 669–676.

Gruenwald, L.; Sadik, M. S.; Shukla, R.; and Yang, H. 2010. Dems: a data mining based technique to handle missing data in mobile sensor network applications. In *Proceedings of the Seventh International Workshop on Data Management for Sensor Networks*, 26–32.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Längkvist, M.; Karlsson, L.; and Loutfi, A. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters* 42:11–24.

Lee, D. D., and Seung, H. S. 2001. Algorithms for non-negative matrix factorization. *Neural information processing systems (NIPS)* 556–562.

Lee, D.; Kulic, D.; and Nakamura, Y. 2008. Missing motion data recovery using factorial hidden markov models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1722–1728.

Lindstrom, J.; Szpiro, A. A.; Sampson, P. D.; Oron, A. P.; Richards, M.; Larson, T.; and Sheppard, L. 2014a. A flexible spatio-temporal model for air pollution with spatial and spatio-temporal covariates. *Environmental and Ecological Statistics* 21(3):411–433.

Lindström, J.; Szpiro, A. A.; Sampson, P. D.; Oron, A. P.; Richards, M.; Larson, T. V.; and Sheppard, L. 2014b. A flexible spatio-temporal model for air pollution with spatial and spatio-temporal covariates. *Environmental and ecological statistics* 21(3):411–433.

Malhotra, P.; Vig, L.; Shroff, G.; and Agarwal, P. 2015. Long short term memory networks for anomaly detection in time series. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*.

Manyika, J.; Chui, M.; Brown, B.; Bughin, J.; Dobbs, R.; Roxburgh, C.; and Byers, A. H. 2011. Big data: The next frontier for innovation, competition, and productivity. *Analytics*.

Mei, H., and Eisner, J. 2017. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems (NIPS)*.

Neil, D.; Pfeiffer, M.; and Liu, S.-C. 2016. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems (NIPS)*, 3882–3890.

Pan, L., and Li, J. 2010. K-nearest neighbor based missing data estimation algorithm in wireless sensor networks. *Wireless Sensor Network* 2(02):115.

Parveen, S., and Green, P. 2004. Speech enhancement with missing data techniques using recurrent neural networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, I–733.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems (NIPS)*, 91–99.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. *International world wide web conferences (WWW)* 285–295.

Sorjamaa, A.; Lendasse, A.; Cornet, Y.; and Deleersnijder, E. 2010. An improved methodology for filling missing values in spatiotemporal climate data set. *Computational Geosciences* 14(1):55–64.

Su, X., and Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence* 4.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning (ICML)*, 1139–1147.

Valipour, M.; Banihabib, M. E.; and Behbahani, S. M. R. 2013. Comparison of the arma, arima, and the autoregressive artificial neural network models in forecasting the monthly inflow of dez dam reservoir. *Journal of hydrology* 476:433–441.

Wu, T., and Li, Y. 2013. Spatial interpolation of temperature in the united states using residual kriging. *Applied Geography* 44:112–120.

Yi, X.; Zheng, Y.; Zhang, J.; and Li, T. 2016. St-mvl: filling missing values in geo-sensory time series data. In *International joint conference on artificial intelligence (IJCAI)*, 2704–2710.

Yozgatligil, C.; Aslan, S.; Iyigun, C.; and Batmaz, I. 2013. Comparison of missing value imputation methods in time series: the case of turkish meteorological data. *Theoretical and applied climatology* 112(1-2):143–167.

Yu Zheng, Furui Liu, H.-P. H. 2013. U-air: When urban air quality inference meets big data. *Knowledge discovery and data mining (KDD)* 1436–1444.