A Collaborative Filtering Approach to Estimating Missing Sensor Data

Abstract

Data sets gathered from sensor networks often suffer from a significant fraction of missing data, due to issues such as communication interference, sensor interference, power depletion, and hardware failure. Many standard data analysis tools such as classification engines, time-sequence pattern analysis modules, and even statistical tools are ill-equipped to deal will missing values—hence, there is a need to impute missing readings prior to analysis.

In this paper, we present novel imputation methods that take a "Recommendation Systems" view of the problem: the sensors and their readings at each time step are viewed as products and user product ratings, with the goal of estimating the missing ratings. Sensor readings differ from product ratings, however, in that the former exhibit high correlation in both time and space. To incorporate this property, we modify the widely successful Matrix Factorization (MF) approach for recommendation systems to model temporal correlations and learn latent relationships among sensors. We evaluate the approach using two environmental sensor network datasets, one indoor and one outdoor, and two imputation tasks, corresponding to intermittent readings and failed sensors. The results show that our Temporally-regularized MF (TRMF) approach provides significantly higher estimation accuracy than both (i) state-of-the-art recommendation models and (ii) state-of-the-art sensor data imputation approaches such as the hybrid-KNN model. Interestingly, adding spatial coordinate information into TRMF is shown to be ineffective—TRMF already captures the latent relationships among sensors, including spatial correlations.

Next, we consider sensor networks with multiple sensor types at each node. We present two techniques for extending TRMF to account for possible correlations among sensor types (e.g., temperature and humidity): Multivariate TRMF and Temporally-regularized Tensor Factorization. Our results show that both techniques are significantly more accurate than prior approaches, and each has its strengths, depending on the observed variance in the readings. Finally, we consider a popular data analysis task—building regression-based prediction models—and show that, compared to prior approaches, applying our more-accurate imputation techniques leads to higher-quality prediction models.

Keywords

missing data recovery, matrix factorization, tensor factorization, temporal regularization, multivariate learning, data imputation

Note: Neither the abstract nor the introduction have been updated to reflect our recent findings that spatial regularization can help for the temporal split case.

1 Introduction

Wireless sensor networks (WSNs) are especially susceptible to interference, battery depletion, and other environmental and communications ailments which lead to data loss. Data sets gathered from sensor networks are often missing 10%–40% of the possible readings, depending on the severity of the environmental factors [**?**]. These missing values are problematic for data analysis tools such as classification engines, time-sequence pattern analysis modules, and other machine learning tasks, which are often ill-equipped to deal with missing values. Support Vector Machine (SVM) [**?**] and Multiple Regression (MR) analysis, to name but a few examples, require complete datasets with no missing values. Popular statistical packages such as SAS, Stata, and R provide a few default options for handling missing data, as a preprocessing step, because the core algorithms need all data filled in. Typical options are (i) remove the entire row if there is a missing value, or (ii) fill in the missing value (called *imputation*) using either simple defaults like the average of neighboring values or user-written code. The first option discards Backgroundotherwise useful data, and in fact, may discard most of the rows in datasets with high data loss! Thus, imputation is a vital tool in the preparation of sensor data for subsequent analysis. Because the accuracy of the target data analysis depends on the accuracy of the imputation, improvements in sensor data imputation can better serve sensor network deployment objectives.

We consider the common setting where the goal is to collect all the sensor readings in order to perform centralized analysis, while maximizing the lifetime of the WSN. Sensor nodes are battery powered and may be energy-harvesting,

SenSys'12, November 6–9, 2012, Toronto, ON, Canada.

and will often make only a "best effort" attempt to transmit their readings back to the centralized collection point. This contributes to the prevalence of missing data, furthering the value of effective imputation techniques.

1.1 Existing Imputation Techniques

Imputation techniques as applied to sensor data can be divided into three categories: temporal methods (i.e., estimation using the observations from the target sensor at nearby time-steps), spatial methods (i.e., estimation using neighboring sensor node observations), and hybrid spatio-temporal methods, as shown in Table 1.

Temporal methods leverage the temporal correlation among readings by the same sensor node; salient methods include observed data mean [**?**, **?**], last seen [**?**], and linear interpolation. These methods suffer, however, when there are long temporal gaps of data for a given sensor; such gaps can be frequent in WSNs due to power depletion in energy-harvesting sensors, communication ailments that last multiple time steps, etc. As a result, the usefulness of temporal imputation methods drop rapidly as the number of consecutively missing readings becomes large.

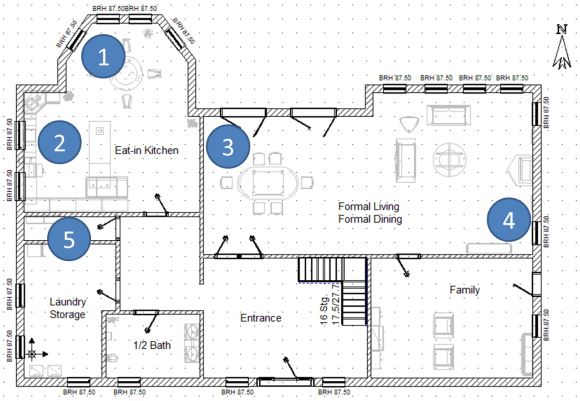


Figure 1. Example home floorplan showing five deployed temperature sensors

Spatial methods leverage the spatial correlation among readings by nearby sensor nodes; salient methods include associations rule mining (e.g., WARM [**?**] and FARM [**?**]) and weighted functions of nearby sensors (e.g., DEPM [**?**], K-NN [**?**], and Multi-Im [**?**]). These methods require reasonably accurate spatial coordinates, and more importantly, suffer when there are ailments that affect entire spatial regions such as may arise in the presence of a large obstacle to sensing and/or communication. Spatial methods also fail to take into account barriers or other sources of sharp environmental gradients that may deter the usage of spatial information as a first-order inter-sensor node correlation approximation. For example, in Figure 1, we find sensors 1 & 2 deployed in the kitchen, nearby the stove and outside window, respectively. While these sensors are close in proximity, assuming the stove is in use and the temperature outside is cold, there may be a large temperature difference between these two sensors despite their close proximity. Similarly, sensors 2 & 5 may be quite uncorrelated despite their relative proximity due to the wall between them and the presence of kitchen or laundry appliance use. On the other hand, sensors 3 & 4, while located further from one another may be quite correlated despite their remote placement as they are both near an outside wall and both within the same room. The calculation of inter-node signal strength or line of sight distance between nodes can help to mitigate the issues of spatially-based imputation, though it is not a complete remedy. In the end, the incorporation of spatial information can lead to worse imputation results as non-existent (or at least inconsistent) correlations are imposed between sensors.

Certain methods consider not strictly the distance between sensors, but instead establish a "neighborhood of influence" whose size becomes a tuning parameter of this approach, which adds to the complexity of this approach.

Finally, *hybrid spatio-temporal methods* consider both the temporal and spatial correlation; salient methods include STI [**?**], DESM [**?**], AKE [**?**], and Imputation Method [**?**]. These have the potential advantage of using both types of correlation in imputation, but can suffer from the spatial correlation issues discussed above.

1.2 Our Approach: Collaborative Filtering

In this paper, we employ a novel collaborative-filtering (CF) approach to sensor data imputation inspired by the field of Recommendation Systems. In typical CF approaches, the elements of interest are users and items (e.g., products), and the values are user ratings of those items (as in the left-hand side of Figure 2). Typically, most of the ratings are missing, and the goal is to predict (impute) the missing ratings in order to "recommend" items to users. By viewing sensors as items, users as time steps, and readings as ratings (as illustrated in Figure 2), we can apply CF techniques to perform sensor data imputation. (Alternatively, sensors can be viewed as users and items as time steps—the mapping is irrelevant to the CF formulation.) In particular, we focus on the widely successful *Matrix Factorization* (MF) technique for CF.

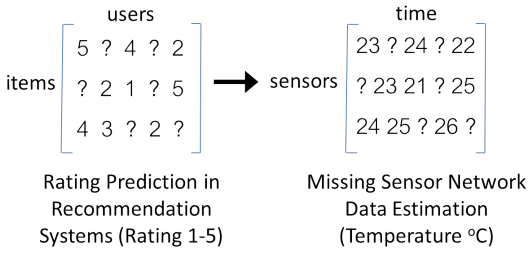


Figure 2. Bridge from Recommendation Systems to Sensor Data Imputation

Sensor readings differ from user ratings, however, in that the former exhibit high correlation in both time and space (subject to the above caveats on space). To incorporate this property, we first modify MF to model temporal correlations and learn latent relationships among sensors. Specifically, we add a *temporal regulation* bias to MF—we call this *temporally-regulated MF* (TRMF).

Second, we consider sensor networks with multiple sensor types at each node. We are able to exploit such heterogeneous sensor information in our solution, which few other

Table 1. Salient Methods for Sensor Data Imputation

Collection	Hot-Deck Imputation	Prediction Models
Temporal	Last-seen [**?**] Mean	Linear Interpolation
Spatial	WARM [**?**] FARM [**?**]	DEPM [**?**] K-NN [**?**] Multi-Im [**?**]
Hybrid Spatio-Temporal	STI [**?**]	DESM [**?**] AKE [**?**] Imputation Method [**?**]

methods have proposed a proper way to incorporate. Our method incorporates these heterogeneous sensor signals (for example, estimating the temperature at a given sensor node utilizing the humidity, temperature, and light trends from other sensors in the network) to provide more accurate imputation than prior approaches. We present two techniques for extending TRMF to account for possible correlations among sensor types: *Multivariate TRMF* and *Temporally-regularized Tensor Factorization* (TRTF).

We evaluate our approaches using two environmental sensor network datasets, one indoor and one outdoor. Both datasets record temperature, humidity, and light within its deployed environment. Each dataset has an initial missing rate (strengthening the claim that missing data in WSNs is a common issue), to which we additionally cover known observations to use for validation and testing purposes. We study two patterns for missing data: (i) covering random readings (modeling intermittent reading failures) and (ii) covering all readings for a subset of sensor nodes from a particular time onwards (modeling long temporal gaps such as with failed sensors).

Our results show that TRMF provides significantly higher estimation accuracy than both (i) state-of-the-art recommendation models and (ii) state-of-the-art sensor data imputation approaches such as the hybrid-KNN model. Interestingly, adding spatial coordinate information into TRMF is shown to be ineffective—TRMF already captures the latent relationships among sensors, including spatial correlations.

For the heterogeneous setting, our results show that both TRTF and Multivariate TRMF are significantly more accurate than prior approaches, and each has its strengths, depending on the observed variance in the readings. Finally, we consider a popular data analysis task—building regression-based prediction models—and show that, compared to prior approaches, applying our more-accurate imputation techniques leads to higher-quality prediction models.

These results validate our novel approach of equating sensor imputation with recommendation systems. CF approaches such as Matrix Factorization and Tensor Factorization are adept at handling sceanarios with large numbers of missing values. Correlations are captured by grouping correlated sensor nodes and correlated time steps—unlike prior sensor data imputation approaches, our CF approaches use this *latent* information to impute values. Moreover, our CF approaches are global, taking into account all collected observations, and not overly tied to distance-based spatial correlations. For example, they can capture the correlations between distant sensors 3 & 4 in Figure 1, while grouping sensors 1, 2 & 5 only if the observed readings warrant it. The CF framework also provides a unified approach to incorporate any number of additional sensor types for even more accurate imputation.

1.3 Contributions

In summary, the main contributions of this paper are:

- We propose viewing sensor data imputation as a recommendation systems problem, and apply modern collaborative filtering methods of recommendation systems (namely, matrix and tensor factorization) to the sensor network domain.
- We augment collaborative filtering with temporal regularization and multi-sensor signals, and provide efficient optimization methods to learn the inherent model parameters effectively.
- We present an empirical study on two sensor datasets, considering two imputation tasks that correspond to intermittent readings and failed sensors. The results show that our proposed approaches provide significantly higher estimation accuracy than state-of-the-art prior approaches, and moreover, such accuracy improvements can result in the generation of higher-quality prediction models.

1.4 Paper Organization

The remainder of our paper is organized as follows. Related Work is reviewed in Section 2. In Sections 3 and 4 we describe our Matrix and Tensor Factorization approaches to WSN data imputation, respectively. Sections 6 and 7 provide discussion of our findings and the conclusion.

2 Related Work

Missing data imputation is the process used to fill the missing data items by determine or assign values [**?**]. The physical characteristic of WSNs has shown that there are strong spatial and temporal correlation between sensor nodes and its past historical readings [**?**]. Sensors, according to their physical phenomenon, have multivariate correlations [**?**]. They build a probabilistic model to estimate the missing values exploiting spatial, temporal, and multivariate correlation conditioned on the observations.

According to the strategy deployed to fill in the missing values, sensory data imputation method can be partitioned into hot-deck imputation, and prediction models [**?**]. Hot-deck imputation methods directly replace the value from either neighbourhood values or historical record from itself

such as the *last-seen* method. The prediction models provide a function to estimate the missing value using the historical data. Table 1 summarize the sensory data imputation methods. The vertical dimension in Table 1 shows whether it exploits the temporal, spatial, or hybrid hybrid of both information. The horizontal dimension classifies the methods by how the missing value is produced, either direct substitution or through prediction.

2.1 Predictions based on Temporal Information

Linear interpolation(LIN) is a common temporal correlation based technique for missing data prediction. It is usually regarded as a baseline method, and has been implemented into some data analysis tools. Mathematically, the estimated value $\hat{y_{it}}$ is: $\hat{y_{it}} = y_{iu} + \frac{y_{iv}-y_iu}{T_v-T_u}(t-T_u)$. $y_{iu}$and yiv are the previous and the next observations for sensor i at time t. Linear interpolation method does not considers correlation among sensors. In general, its performance is acceptable when the missing data are scattered. However, the quality of imputation goes down significantly if the sampling rates is low or there are consecutive data missing. Imputation based on the last-seen, moving average, and other statistics methods apply on the target sensors generally belong to the temporal-based method as well. These methods share the advantage of comprehensible and easy implementation. However,their performance depends on the sensor data missing pattern as well as sampling rates.

2.2 Prediction based on spatial information

The physical phenomenon of spatial correlation in sensors measurement can be divided into two categories, the point source and the field source [**?**]. For point source sensors, the relation between the sensors in a region satisfies physical laws in some event detecting sensors such as light ambient sensors and fire detector. Their prediction values are the result of linear superposition by neighbourhood sensors. Data Estimation using Physical Model(DEPM) [**?**] employs the basic law of Physics. They set up the prediction model, $I_K = \sum_{j=1}^{M} \frac{P_j}{4\pi d^2(I_j,s_k)}$ where I_k is the intensity of target sensor, j is the neighbour sensors, and P stands for the power radiated from two sources. However, it requires the precise three-dimensional distance between two sensors, therefore it is not realistic in large sensors deployment.

Researchers also propose to predict missing values based on the data mining techniques. Window Association Rule Mining(WARM) [**?**] and Freshness Association Rule Mining(FARM) [**?**] studied the estimations of missing data based on the association rules between spatially correlated neighbours. It has the advantage of handling categorical sensor data, but the performance is limited in continuous sensor data streams due to the limitation of association rules. Moreover, the support, confidence of the mined rules need to be predefined by the users which can cause difficulties for building a satisfying model by users without profound knowledge about the environment of deployment.

The multiple imputation(MI) procedure [**?**] imputes the missing data by replaces each missing value with a set of plausible values instead of filling in a single value. Various methods including multiple linear regression, propensity score method, and Markov chain Monte Carlo method are are reported and used in MI. The missing data are filled in M times(M=3-10, [**?**]) to generate M complete data sets, and the results from all the M generated data sets are combined for the inference [**?**]. However, this method is critized on various grounds by WSNs imputation researchers [**?**] [**?**]. First, in sensor data stream, we do not know how many rounds of information shall we use in order to get the associated information. In addition, it is difficult to draw a pool of similar complete cases for a certain round of a certain sensor. Last, it consumes unnecessary time since the sensor data may or may not related to all of the available information.

2.3 Hybrid Spatio-temporal Imputation

According to past studies, algorithms that only use either spatial data or temporal data inevitably face problems [**?**]. In the environment when the radio transmission quality is poor or the packets collision happens frequently, exploiting only temporal information such as the linear interpolation is not ideal, like the linear interpolation. In contrast, for certain sensing applications like light sensor(point source), the spatial information is often unreliable since more than one light source can be sensed. The predictor using only spatial data can easily be harmed by the bias and drift from neighbourhood sensing data. Therefore, applying both temporal and spatial information on the sensory missing data problem seems inevitable. However, the methods that use spatial correlation requires the knowledge of mutual distance between sensors.

An algorithm that uses spatial and temporal imputation(STI) is proposed in [**?**]. When a packet is missing, the STI first checks if a neighbour node is within the missing sensor's sensing region. The average of the neighbours in the range is then used to impute the missing value. Otherwise, the last seen value of the missing sensor is used for prediction. The method prefers spatial prediction more than temporal. It has the advantage of simple, fast, and little need of extra memory in the restricted computing resources in WSNs. However, problems can arise when the last seen value is far away from the current time stamp, and when most of the nodes in the sensing region has high missing rate.

Another algorithm which replying more on spatial more temporal is the Data Estimation using Statistical Model (DESM) [**?**]. The prediction value of sensor i is: $\hat{y_{it}} = (1-\alpha)\hat{y_{i(t-1)}} + (\alpha)\hat{z}$ whereα is the Pearson correlation coefficient between the sensor i and the nearest sensor j. The above equation assumes that the X_i and X_j have the similar data fluctuation trend which is true if they are very close to each other. α is expected to have more impact on the prediction if X_i is more correlated with X_j. $\hat{y_{it}}$ is the last estimated value, which measures the influence of the historical sensing data on the current value at node i. $\hat{z}$ is the estimation of sensor j based on the observation from sensor i at the time m. $\hat{z} = X_{j(m)}(1 + \frac{X_{i(m+1)}-X_{(im)}}{X_{i(m)}})$ It measures the influence of the data sensed by the active node i on the data of the nearest node j. In the method, the temporal feature holds a very small portion of contribution to the prediction process and it often contradicts the nature that the temporal correlation it-

self is usually higher than any spatial correlation with other sensors. The performance may becomes bad if the nearest sensor is not the most correlated one, i.e. blocked by a wall.

K-nearest neighbours algorithm(KNN) is a intuitive spatial correlation based imputation method. It has been adopted to estimate the missing value of DNA micro-arrays [**?**]. However, it only directly uses the weight average of other genes to fill the missing data. While in WSNs, the sensor data of different nodes is more likely to have some functional relations other than just using sensor values. Pan [**?**] propose the Applying K-nearest neighbour Estimation algorithm(AKE) to exploit the spatial correlation in the missing sensory data problem. They argue that the past research on sensor imputation(STI) estimates the missing problem based on more temporal correlation than spatial correlation(STI). In their observation, the sensor data in WSN often changes rapidly and sharply especially in outside environment. The AKE adopts the linear regression model, $\hat{y}_{it} = \hat{\alpha} + \hat{\beta} \cdot y_{jt}$ where *it* means the time t of sensor i, and j is its neighbour nodes. They claim that using a linear combination of the estimation from neighbour sensors can lower the random error caused by only using a single sensor. For given node i whose data is missing at time t, and the m nearest neighbour sensors who have values at time $t[y_{1t}, y_{2t}, \ldots, y_{mt}]$, the missing value is predicted by :

$$\hat{y}_{it} = \sum_{j=1}^{m} w_j \cdot \hat{y}_{it}^{(j)} \tag{2.1}$$

In eq2.1, the estimated $\hat{y}_{it}^{j}$ is computed by node j using the coefficients of $[i, j]$. In designing the weighting function, the AKE gives higher correlation pairs a larger weight. They adopts the r-square statistics to rank their correlations due to when sensors have the higher R^2, they have the better regression equation fit. However, in some environment, the distance used to decide the m nearest neighbours is not feasible. Moreover, in certain scenarios where two far away sensors have higher correlation, i.e. two sensors place under the same air-conditioning tuyere, the AKE will ignore the far away sensor. Finally, the preference on spatial over temporal is also being challenged when the sampling rates is very high, i.e. the body sensors, using temporal linear interpolation has better prediction.

In [**?**], an imputation method(ImM) consists of two temporal and one spatial predictors was introduced. The algorithm argues that it is not robust to set the preference for its spatial predictor over temporal predictor, or vice versa. They assume that the predictor which yields the most accurate results in the training data set is used to predict the missing packets. However, the performance of the ImM may degrades if the changes in the continuous data stream are sharp in the high sampling rate data set. For example, the wind sensors in the windy climate area may affect the method applied on the target sensor.

2.4 Review of Matrix Factorization

Matrix factorization has many applications in computer visions, recommendation system [**?**], and data imputation. Their goals are to factorize the measurement matrix R and find the two subspace: U and V, $M_{d\times n} = U_{d\times k} V^t_{k\times n}$. We will introduce two applications other than WSNs data imputation in the following sections. The first application is the EOF-based prediction method in satellite photography and the second one is the Structure-from-motion(SFM) in computer vision.

Empirical Orthogonal Functions(EOF) has been applied to oceanographic application to solve the problem of missing or unreliable satellite data [**?**]. They propose the DINEOF (Data Interpolating Empirical Orthogonal Functions) which is an Singular value decomposition(SVD)-based technique for the reconstruction of missing data. In their method, given a matrix X_{m*n} with its entry $(X)_{ij}$ which contains the observations values at i and moment j. U_N is the matrix constructed by the first N spatial EOFs, which have the same meaning as N-modes in SVD. At the beginning, the missing values are filled with zero first, then the prediction is by: $(X_\gamma)_{ij} = (U_N D_N V_N^T)_{ij}$ The operations stop when X_{ij} is filled by all the missing item with their prediction in $(X_\gamma)_{ij}$. Researchers of [**?**] point out that when large gap of data missing due to clouds or rains, the remained observations may not be representative of the whole spatial variability of the domain. Therefore they apply a Laplacian filter F on the terms of X before the estimations of EOFs. After filtering, two consecutive images in matrix X with a large time gap between them are less related than two consecutive images. However, there are several drawbacks in the SVD-based methods when apply to impute large-scale sensors network. First, SVD needs to fill all the missing values before factorization which can be very expensive as it significantly increases the amount of data. The inaccurate imputation might also distort the data considerably [**?**]. In addition, [**?**] point out SVD is sensitive to outliers and missing data.

Structure from motion(SFM) [**?**] and motion estimation are among the most important application in computer vision. A number of studies have investigated different approaches to the MF-based structure from motion problem. [**?**] performs the L2-norm minimization using the standard Newton algorithm with a damping factor. In [**?**], they propose a new method incorporates the damping factor into the Wiberg method to solve the L2-norm problem. A L1-norm factorization using Wiberg method is proposed in [**?**]. However, the size and ranks of the video data is different from the long-term and densely deployed sensor data. The rank of linear subspace in SFM has rank of three after it is subtracted by its column-wise mean. It is also noted that the video camera in essential is an univariate sensor device which is different from the nature of heterogeneity of sensor networks.

3 Matrix Factorization Model for Imputation

Matrix Factorization (MF) is arguably the most successful Collaborative Filtering (CF) techniques in the area of Recommendation Systems [cite]. Comparing to other recommendation model such as regression-based prediction models, graph-based random walk models, or simple statistic models, MF model possesses the edge of being more accurate and scalable to data size. The key advantage of MF models lies in the capability to learn latent factors from the relatively sparsed observation, and leverage these factors to reproduce the missing elements in the matrix. In the fol-

lowing, we will first introduce the fundamentals of the MF methodology, communicate our novel sensor network data specific modifications to the MF objective function, and finally provide the complete training procedure of the proposed method.

3.1 Introducing Matrix Factorization

Matrix Factorization is designed to amend the limitations of a widely known dimension reduction technique called Singular Value Decomposition (SVD) in dealing with incomplete or sparse data. The goal of SVD is to decompose a fullly observed matrix $\mathbf{R}$ into one diagonal matrix $\mathbf{D}$ and two unitary matrices $\mathbf{U}$ and $\mathbf{V}$ such that

$$\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^T.$$

The largest K singular values, $\mathbf{U}_K\mathbf{D}_K\mathbf{V}_K^T$, form the best K-rank approximation of $\mathbf{R}$ under the Frobenius Norm. It has been shown that given a complete matrix, SVD can identify the k-latent features that minimizes the L2 error. Unfortunately, SVD cannot be performed when the matrix $\mathbf{R}$ is incomplete. Therefore, to exploit latent dimensions through SVD for imputation, the ealier works have to fill in missing entries first (usually using zeros or an averaged value) and then perform dimensions reduction through multiplying the matrixes of the k-largest sigular values [cite]. Such strategy does create some dilemma as a high-quality imputation model is needed in the first stage to guarantee the quality of data reproducing in the later stage, but it is such high-quality we are looking for eventually for data imputation. In a nutshell, SVD model, although proven to be ideal for dimension reduction, can hardly be applied to sparse matrix where decent amount of data are missing. To address such drawback, recently researchers have shown [**?**] that the matrix factorization model is indeed a better approach to learn the latent factors given sparse matrixes, because during the factorization procedure only the observed entries are exploited. Utilizing numerical optimization procedures, for a partially observed matrix $\mathbf{R}$, MF produced two latent matrices $\mathbf{P}_{MxK}$ and $\mathbf{Q}_{KxN}$ whose multiplication tries to approximate the observed entries in $\mathbf{R}_{MxN}$:

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}.$$

Given $\mathbf{R}$ being the sensor network readings, each row of $\mathbf{P}$ represents latent factors in the temporal dimension and each column of $\mathbf{Q}$ represents the latent factors in the dimension of correlation among sensors.

We adopt the biased-MF which includes row and column biases μ_m and μ_n. In a temperature monitoring system, the row bias can be understood as the average temperature at a given time, and the column bias reflects the average temperature at the location plus the systematic bias of the sensor node. The predictions of missing values can be obtained through $\hat{r}_{m,n} = \mu_m + \mu_n + \mathbf{p}_m\mathbf{q}_n$. After adding the regularization term to constraint the scale of latent features, the objective function of MF becomes:

$$\frac{1}{2}\sum_{m,n}(r_{m,n}-\hat{r}_{m,n})^2 + \frac{\beta_1}{2}\sum_m \mu_m^2 + \frac{\beta_2}{2}\sum_n \mu_n^2 + \frac{\beta_3}{2}\sum_m ||\mathbf{p}_m||^2 + \frac{\beta_4}{2}\sum_n ||\mathbf{q}_n||^2.$$

, where $\mathbf{p}_m$ are the row factors of $\mathbf{P}$ (for time m), and $\mathbf{q}_n$ are the column factors of $\mathbf{Q}$ (for sensor node n) respectively. β_1, β_2, β_3, β_4 are parameters that control the strength of regularization.

3.2 Temporally-Regularized Matrix Factorization

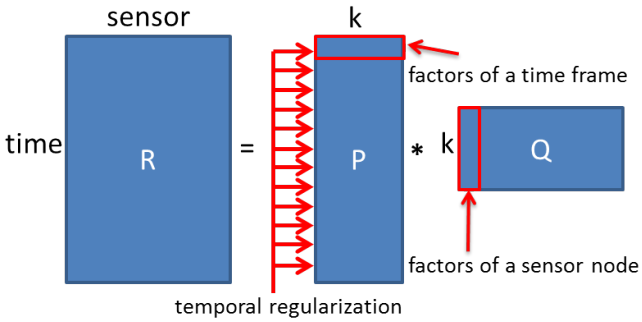


Figure 3. illustration of TR-MF

Although we analogize the WSN data recovery task as a CF-based recommendation task, there are indeed some major differences in terms of the property of data. Normal CF or MF models assume the users (i.e. temporal dimension) are indepent of each other. That is, we can randomly swap the columns in the matrix without affected the factorization outcome. However, such independency does not exist in WSN data as a sensor's signal in time t is highly dependent on that of time t-1. In other words, an ideal model should consider such order dependency, and under such model the swap of columns shall significantly affect the outcome of factorization.

With this observation, we propose a Temporally-Regularized Matrix Factorization (TR-MF) to better model the characteristic of WSN data.

As the name may suggest, TR-MF adds a temporal regularization term to conventional MF. The temporal regularization forces the latent factors of adjacent rows to be similar, which reflects the fact that that readings in adjacent time steps should be similar. We also add similar regularization term for row bias. The modified objective function looks like:

$$\frac{1}{2}\sum_{m,n}(r_{m,n}-\hat{r}_{m,n})^2 + \frac{\beta_1}{2}\sum_m \mu_m^2 + \frac{\beta_2}{2}\sum_n \mu_n^2 + \frac{\beta_3}{2}\sum_m ||\mathbf{p}_m||^2 + \frac{\beta_4}{2}\sum_n ||\mathbf{q}_n||^2 + \frac{1}{2}\gamma_1\sum(\mu_m-\mu_{m+1})^2 + \frac{1}{2}\gamma_2\sum||\mathbf{p}_m-\mathbf{p}_{m+1}||^2.$$

3.3 Spatio-Temporal-Regularized Matrix Factorization

Although previously we argue that near-by sensors might not necessary possess the higheset correlation with each other, here we would like to show that the previously proposed MF model can easily be adopted to accomendate spatial correlation if one decides to do so. Given the distance

(or any kind of 'closeness' measure) between sensors, we can add spatial regularization terms to strengthen the spatial correlation. Therefore, this model enforces sensors closer to each other to have similar sensor measures. The objective function then becomes:

$$\begin{aligned}&\frac{1}{2}\sum_{m,n}(r_{m,n}-\hat{r}_{m,n})^2+\frac{\beta_1}{2}\sum_m\mu_m^2+\frac{\beta_2}{2}\sum_n\mu_n^2\\&+\frac{\beta_3}{2}\sum_m||\mathbf{p}_m||^2+\frac{\beta_4}{2}\sum_n||\mathbf{q}_n||^2\\&+\frac{1}{2}\gamma_1\sum(\mu_m-\mu_{m+1})^2+\frac{1}{2}\gamma_2\sum||\mathbf{p}_m-\mathbf{p}_{m+1}||^2\\&+\frac{1}{2}\gamma_3\sum(\mu_{n_i}-\mu_{n_j})^2+\frac{1}{2}\gamma_4\sum||\mathbf{p}_{n_i}-\mathbf{p}_{n_j}||^2.\end{aligned}$$

We call this Spatio-Temporal-Regularized Matrix Factorization (STR-MF). Note that we suggest users to exploit spatial regularization with case, as will be shown in our experiment section that forcing spacial correlation can sometimes produce inferior results.

3.4 Optimization Procedure

Several methods to learn MF have been proposed, such as Stochastic Gradient Descent (SGD) [**?**, **?**], Alternating Least Square (ALS) [**?**, **?**], Newton's method [**?**] and Wiberg Algorithm [**?**]. For WSN data, we suggest SGD for its efficiency and simplicity.

In SGD, we incrementally update our model by considering one reading at a time. Focused on one observed reading $r_{m,n}$ with the following objective function

$$\frac{1}{2}(r_{m,n}-\hat{r}_{m,n})^2+\frac{\beta_1}{2}\mu_m^2+\frac{\beta_2}{2}\mu_n^2+\frac{\beta_3}{2}||\mathbf{p}_m||^2+\frac{\beta_4}{2}||\mathbf{q}_n||^2.$$

It is not hard to derive the update equations as

$$\begin{cases}\mu_m'=\mu_m-\eta_1((\hat{r}_{m,n}-r_{m,n})+\beta_1\mu_m)\\\mu_n'=\mu_n-\eta_1((\hat{r}_{m,n}-r_{m,n})+\beta_2\mu_n)\\\mathbf{p}_m'=\mathbf{p}_m-\eta_2((\hat{r}_{m,n}-r_{m,n})\mathbf{q}_n+\beta_3\mathbf{p}_m)\\\mathbf{q}_n'=\mathbf{q}_n-\eta_2((\hat{r}_{m,n}-r_{m,n})\mathbf{p}_m+\beta_4\mathbf{q}_n)\end{cases}$$

For each reading, we update the all μ_m and $\mathbf{p}_m$. Then after a full scan of every observed readings, we perform temporal regularization by updatig all μ_m and $\mathbf{p}_m$ simultaneously according to the following equations:

$$\begin{cases}\mu_m'=\mu_m-\eta_1\gamma_1((\mu_m-\mu_{m-1})+(\mu_m-\mu_{m+1}))\\\mathbf{p}_m'=\mathbf{p}_m-\eta_2\gamma_2((\mathbf{p}_m-\mathbf{p}_{m-1})+(\mathbf{p}_m-\mathbf{p}_{m+1}))\end{cases}$$

The updating procedure is summarized in Procedure 1. Note that we propose to avoid updating the temporal regularization with the update of each reading, because doing so can bias the model toward the time steps that possess fewer missing readings, which contradicts the goal of data imputation.

If the spatial regularization is exploited, we can update all μ_n and $\mathbf{q_n}$ simultaneously as:

$$\begin{cases}\mu_{n_i}'=\mu_{n_i}-\eta_1\gamma_3\sum_{n_j}(\mu_{n_i}-\mu_{n_j})\\\mathbf{q_{n_i}}'=\mathbf{q_{n_i}}-\eta_2\gamma_4\sum_{n_j}(\mathbf{q_{n_i}}-\mathbf{q_{n_j}})\end{cases}$$

Data Normalization

Unlike the ratings in recommendation which normally are within certain range (e.g. 1 star to 5 star), readings from WSN are real-valued, and the range may vary with the sensor locations. Here we propose to normalize the training set to N(0,1) before conducting MF learning, and once the missing values are produced by our model, we need to rescale the values to the original mean and variance. Although this procedure does not change the quality of outcomes theoretically, in practice we do find some benefits: First, when the global mean becomes zero, the origin of our model natually becomes a fine initial point for MF training, which is very important for non-convex optimization techniques such as MF. Secondly, normalization forces different data sets to look similar, which simplifies the parameter tuning task for (S)TR-MF. As will be shown in the next section, this trick also facilitates parameter tunings in our multivariate models.

Parameters Setting

There are several parameters in our model: conventional regularization for user biases β_1, item biases β_2, user factors β_3, item factors β_4, temporal regularization for biases γ_1 and for factors γ_2, spatial regularization for biases γ_3 and for factors γ_4, learning rate for biases η_1 and for factors η_2, number of factors K, and it can be time consuming to search for a good combination. Here we provide some empirical suggestions to reduce the search space while still obtain results with decent quality: the rule of thumb is to setup parameters of similar physical meanings to be identical. For example: the following setup is what we exploited for the experiments:

$$\beta_1=\beta_2=\beta_3=\beta_4,\gamma_1=\gamma_2,\gamma_3=\gamma_4,\eta_1=\eta_2.$$

Advanced discussion in this aspect will be shown in Section 6.1.

Finally, to determine when to cease updating the MF model, we use the randomly split validation set to learn the stopping criteria. More speicifically, the training stops when the validation error keeps increasing for certain amount of iterations (currently setup to 500).

Procedure 1 (Spatio-)Temporally-Regularized MF

Parameter: β_1, β_2, β_3, β_4, γ_1, γ_2, η_1, η_2, K
Input: training set, validation set

```
Normalize the training set as 𝒟
Initialize μ_m, μ_n, p_m, q_n for all m, n
repeat
    for each observed reading r_{m,n} in 𝒟
        Update μ_m, μ_n, p_m, q_n
    Update μ_m, p_m by temporal regularization
    (Update μ_n, q_n by spatial regularization)
until stopping criterion is met
Output the model for testing set prediction
```

4 Multivariate Factorization Model

A given sensor node may contain multiple sensor types and thus is capable of sensing various aspects of the environment (e.g. temperature and humidity) at the same time. These attributes can potentially be correlated, and being able to take advantage such correlation would result in better imputation quality. This section proposes two models, the Multivariate (S)TR-MF and Tensor Factorization modeles,

to leverage multivariate correlation for missing data recovery.

4.1 Multivariate (S)TR-MF

In (S)TR-MF, as we normally have much longer time-steps than the number of sensors, the latent matrix **P** is much larger than **Q**, and therefore being able to learn a faithful representatin of hidden features in the temporal-dimension is critical.

Assuming there are two types of sensor in one node: temperature and humidity. Then, using (S)TR-MF we can obtain $\mathbf{P}_{tem}$ and $\mathbf{Q}_{tem}$ from temperature matrix $\mathbf{R}_{tem}$, and get $\mathbf{P}_{hum}$ and $\mathbf{Q}_{hum}$ from humidity matrix $\mathbf{R}_{hum}$. These two **P**s are identical in size, and it is not hard to imagine they should be correlated because they both represent the factors of time step m. Therefore, it might be benefitial if we can try to use both sides of information to learn a unified and better $\mathbf{P}_{hum}$. This observation motivates us to design the Multivariate (S)TR-MF.

In Multivariate (S)TR-MF, **R** represents the horizontal concatenation of the temperature matrix $\mathbf{R}_{tem}$ and $\mathbf{R}_{hum}$

$$\mathbf{R} = [\mathbf{R}_{tem} \quad \mathbf{R}_{hum}]$$

, which allows the temperature model and humidity model to share the common **P** matrix, so they merge the similar factors and communicate the observed information with each other. Note that the **Q** matrix for them are different in the (S)TR-MF model, and remain such in multivariate (S)TR-MF.

The learning process of multivariate (S)TR-MF is very similar to that of (S)TR-MF: the objective function and the optimization process remains the same. Yet, there are some important differences: Firstly, for each row (time step), we need two bias terms: one for temperature readings and the other for humidity readings as they natually are biased differently. Furthermore, $\mathbf{R}_{tem}$ and $\mathbf{R}_{hum}$ of **R** must be normalized independently with their own means and variances.

4.2 Tensor Factorization Model

One main concern for the multivariate (S)TR-MF model is that it does not fully exploit the mutual-dependency between the multiple sensor signals in one node. For instance, we did not specify which column in $\mathbf{R}_{tem}$ is correspond to which column in $\mathbf{R}_{hum}$ as coming from the same node. Therefore, here we propose a more complex tensor model to capture such relationship.

Tensor can be regarded as a high-dimentional matrix, and is usually exploited to represent multi-dimentional data. The previously inroduced MF can model only 2-dimensional correlations such as the sensor/time-step readings. With a 3rd-order tensor, we can add one more dimension into the model (e.g. sensor1/sensor2/time-step or sensor/location/time-step). In order to deal with such high-dimensional data structure, mathmaticians have proposed the tensor-decomposition methods to capture its latern features.

Tensor decomposition is a multi-dimensional extension of Singular Value Decomposition (SVD). Similar to SVD, Tensor Decomposition methods assume a fully occupied matrix **R**, while such assumption is not as valid for data imputation purpose as for dimension reduction purpose. In the followings, we will first introduce the tensor decomposition models and than describe how we can modify one of them into a tensor factorization model for missing data recovery.

4.2.1 Tensor Decomposition

Here we introduce two tensor decomposition models in Figure 4. The Tucker decomposition model was first introduced in 1963 [**?**], which factorizes a higher-order tensor into a core tensor **S** and one factor matrix for each dimensions. A $M \times N \times C$ tensor **T** can be decomposed as:

$$\mathbf{T} = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} \mathbf{S}_{ijk} p_i \otimes q_j \otimes w_k$$

or for each component,

$$\mathbf{T}_{mnc} = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} \mathbf{S}_{ijk} \mathbf{P}_{mi} \mathbf{Q}_{nj} \mathbf{W}_{ck}$$

where the vectors p_i, q_j, and w_k are the columns of matrices **P**, **Q** and **W** respectively, which are the factor matrices and usually orthogonal.

Tucker decomposition is computationally expensive, and researchers have proposed a more efficient decomposition named Canonical [**?**]that factorizes a tensor into a sum of component rank-one tensors. Formally, the Canonical Decomposition(CD) is the special case of the Tucker decomposition when **S** is superdiagonal. The Canonical Decomposition of **T** is

$$\mathbf{T} = \sum_{k=1}^{K} x_k \otimes y_k \otimes z_k$$

or

$$\mathbf{T}_{mnc} = \sum_{k=1}^{K} \mathbf{X}_{mk} \mathbf{Y}_{nk} \mathbf{Z}_{ck}$$

where x_i,y_i and z_i are the column of matrices **X**, **Y** and **Z**, which are the factor matrices, K is the factor size.

4.2.2 Tensor Factorization for Data Imputation

Existing Tensor Decomposition models require a dense tensor, which is unsuitable for missing data estimation with a decent missing rate. Therefore, we introduce an N-order Tensor Factorization (TF) model for missing data estimation. Although tensor factorization has been studied for a while [**?**] [**?**] and enjoys certain level of success in building context-aware recommender systems [**?**] [**?**] [**?**], we have not yet seen any proposal to leverage such techniques for WSN data imputation.

Similar to MF, TF learns the temporal correlation given the sparsity of data with the capability to take additional information such as spatial correlation and heterogeneous sensor readings into consideration. Considering each reading as a three-dimensional tensor element (e.g. the temparature reading of a sensor given certain time-stamp and certain humidity reading) in R, the TF model can be described as :

$$\mathbf{T} := \mathbf{R} \rightarrow \mathbf{F_1} \times \mathbf{F_2} \times \mathbf{F_3}$$

Such tensor factorization model decompose the reading tensors into three features F_1, F_2, F_3, one of them represents the temporal dimension, the rest can represent sensor

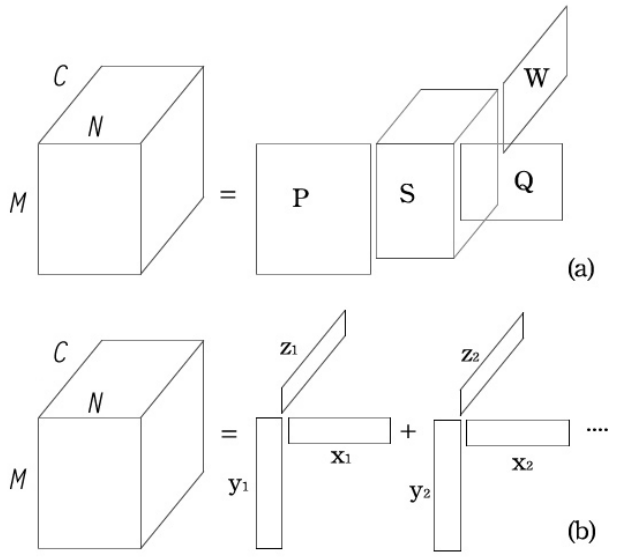


Figure 4. (a) the Tucker Decomposition (TD) model (b) the Canonical Decomposition model, a special case of the TD model.In sensor networks, the three dimension can be nominal features such as node id, time step index.

nodes, sensor node coordinates, heterogeneous sensor readings, etc. In the experiments, we implemented a 3rd-order tensor model and choose the sensor nodes as well as the multivariate sensor readings as the remaining two dimensions.

In contrast to a conventional tensor decomposition, the factorization model we proposed aims at learning the latent factors **P**, **Q**, **W** of the three dimensions F_1, F_2, F_3. To avoid over-fitting and to reduce the complexity of prediction, our model borrows the idea of Canonical Decomposition rather than Tucker decomposition. Note that the time complexity for making prediction is only O(K), where K is the factor size of P, Q and W. Note that the complexity is independent of the rank of the tensor, which is a favorable property because we can freely add new dimensions into the model without concerning about the efficiency.

Similar to the MF model, we also add bias term of each dimension into the TF model. The prediction function is:

$$\begin{aligned}\hat{\mathbf{T}}_{mnc} &= \mu_m + \mu_n + \mu_c + p_m \otimes q_n \otimes w_c \\ &= \mu_m + \mu_n + \mu_c + \sum_{k=1}^{K} \mathbf{P}_{mk}\mathbf{Q}_{nk}\mathbf{W}_{ck}\end{aligned}$$

To learn the latent features P, Q and W, we use the loss function :

$$L(\hat{\mathbf{T}}, \mathbf{T}) = \frac{1}{\|\mathbf{D}\|_1} \sum_{t \in \mathbf{D}} l(\hat{t}, t)$$

where $\|\mathbf{D}\|$ indicates the size of the observed readings. l is a point-wise loss function penalizing the distance between estimate and observation. ideally l should be as close to the eventual evaluation metrics (e.g. least square error in our implementation) as possible.

Simply minimizing a loss function in TF is known to be prone to over-fitting. Here we add regularization terms based on the l_2 norm to ensure that the model complexity does not grow without bound to avoid over-fitting. Similar to our MF model, we also add the time regularization terms into our model. Finally, the objective function of tensor factorization becomes :

$$\begin{aligned}&\sum_{m,n,c} l(\hat{\mathbf{T}}_{mnc}, \mathbf{T}_{mnc}) + \beta_1\|p_\beta\|^2 + \beta_2\|q_n\|^2 + \beta_3\|w_c\|^2 + \beta_4\|\mu_m\|^2 \\ &+ \beta_5\|\mu_n\|^2 + \beta_6\|\mu_c\|^2 + \frac{1}{2}\gamma_1 \sum (\mu_m - \mu_{m+1})^2 + (\mu_m - \mu_{m-1})^2 \\ &+ \frac{1}{2}\gamma_2 \sum (p_m - p_{m+1})^2 + (p_m - p_{m-1})^2\end{aligned}$$

Procedure 2 Multivariate Tensor Factorization

Require: $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \gamma_1, \gamma_2, \eta, k$
1: Normalize the training set as D
2: initial the $p_m, q_n, w_c, \mu_m, \mu_n, \mu_c$ for all m, n, c
3: **repeat**
4: **for** each observed readings t in D **do**
5: Update $p_m, q_n, w_c, \mu_m, \mu_n, \mu_c$
6: **end for**
7: Update p_m, μ_m for all m
8: **until** stopping criterion is met
9: Output the model

4.2.3 Optimization

Minimizing the above objective function can be done using many different strategies. For efficiency and scalability, we suggest using the stochastic gradient descent (SGD) method. We need to compute the gradients of the objective function with respect to each individual component in the model. Focusing on an observed reading data t = $T_{m\beta\gamma}$, the update rules are:

$$\begin{aligned}p_m{}' &= p_m + \eta(e * q_n w_c - \beta_1 p_m) \\ q_n{}' &= q_n + \eta(e * p_m w_c - \beta_2 q_n) \\ w_c{}' &= w_c + \eta(e * p_m q_n - \beta_3 w_c) \\ \mu_m{}' &= \mu_m + \eta(e - \beta_4 \mu_m) \\ \mu_n{}' &= \mu_n + \eta(e - \beta_5 \mu_n) \\ \mu_c{}' &= \mu_c + \eta(e - \beta_6 \mu_c)\end{aligned}$$

where $e = t - \hat{t}$. After a round of updating, we then adjust the parameters according to temporal regularization :

$$\begin{aligned}p_m{}' &= p_m + \eta\gamma_1(p_{m+1} - p_m + p_{m-1} - p_m) \\ \mu_m{}' &= \mu_m + \eta\gamma_2(\mu_{m+1} - \mu_m + \mu_{m-1} - \mu_m)\end{aligned}$$

The Tensor Factorization method is summarized in Procedure 2, which is not hard to implement since it accesses only one row of P, Q ,W at a time.

TF applies similar normalization and stopping criterion trick as shown in section 3. Empirically one can usually achieve reasonable performance by imposing the following constraints on the parameter learning:

$$\begin{aligned}\beta_1 &= \beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 \\ \gamma_1 &= \gamma_2\end{aligned}$$

5 Experimental Results

5.1 Experimental Setup

5.1.1 Datasets

Berkeley Dataset

The Intel Berkeley Research lab dataset [**?**] records temperature, humidity, light, and voltage for 54 sensors (with 1 missing completely missing) for which locations are given (see Figure 5). The dataset includes 2.3M sensor observations, over 210K timesteps, and was recorded between February 28th and April 5th, 2004 in an indoor lab environment.

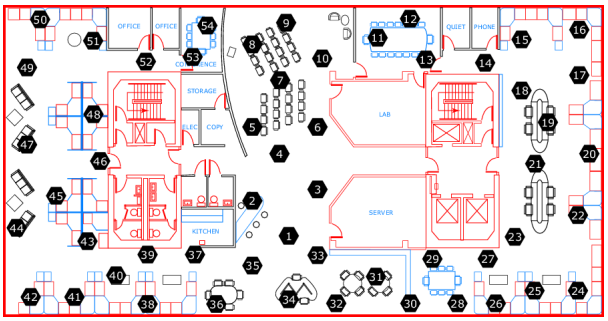


Figure 5. Intel Berkeley Research Lab Floorplan

Traffic Dataset

The Traffic Dataset records the temperature, humidity, and voltage conditions of 20 sensor nodes and one gateway node. This dataset, collected by the Bio-industrial Department at National Taiwan University, was recorded over a 2.5 year time period ending in 2011 in an outdoor location high traffic area in Taipei, Taiwan [**?**].

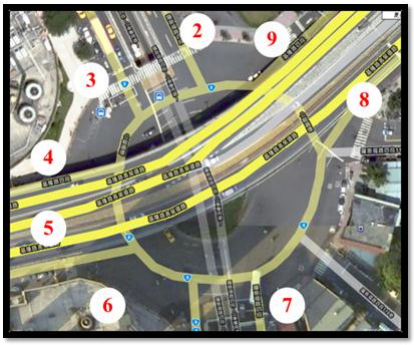


Figure 6. Traffic Sensor Deployment Configuration (7 sensors of 20 shown)

5.1.2 Dataset Preparation

Berkeley Dataset Outlier Removal and Gridding

Outlier Filtering : Observation removed if temperature $> 100\,^{\circ}\text{C}$, temperature $< 5\,^{\circ}\text{C}$, or humidity $< 16\%$.

Gridding : Dataset falls on even 30s intervals for the first 5000 time steps (which is all we consider), so no additional gridding need be performed.

Traffic Dataset Outlier Removal and Gridding

Outlier Filtering : Observation removed if temperature $< 5\,^{\circ}\text{C}$ or $> 60\,^{\circ}\text{C}$ or if humidity $< 10\%$ or $> 100\%$.

Gridding : The original dataset recorded most readings at around the $xx:03$ and $xx:33$ minute marks, so a 6 min window centered at these points captured the data for 30 minute internal readings. Where more than one reading was recorded for a given node within a given window, the closest to the 3 or 33 minute mark was chosen. The full length of the dataset was used, which consists of ≈ 43K time-steps.

Common Methodology

There is an initially missing portion of the observations which can be considered Missing At Random (MAR). To this, we impose two different types of Missing Completely At Random (MCAR) sampling techniques to build validation and testing datasets.

Random Missing Pattern

This pattern reflects choosing a random time and random sensor to be missing and hence removed from the training set. We define two values x and y as follows.

- 10% of the non-missing data is randomly selected (without replacement) to be the validation set
- $y\%$ of the non-missing data is randomly selected (without replacement) to be the testing set
- The remaining data is part of the training set (this would be $x\%$ of the total data given no missing; assuming missing rate of $m\%$, the actual data in training set is $90\% - x\% - m\%$

Temporal Missing Pattern

This pattern reflects testing the effect of all data missing after a certain point in time. We define two values x and y as follows.

- Here, we have the last $y\%$ of time is in test and the prior 10% to test is the validation.
- The sensor node numbers "covered up" in the validation and testing for the Berkeley and Traffic datasets are $4, 19, 45$ and $2, 4, 6, 8, 10, 14, 17, 19, 20, 21$, respectively. Note that node 21 of the Traffic dataset is the gateway node.

5.2 Experimental Results on TR-MF

We compare our TR-MF with conventional MF (i.e. MF without temporal regularization), Linear Interpolation (LI), Applying K-nearest neighbour Estimation (AKE), Data Estimation using Statistical Model (DESM), Spatial temporal imputation (STI) and Multiple Inputation (MI) on the two data sets. We implemented AKE, DESM, STI, and for MI we use the EM-imputation from LISREL. Note that the sensor location information is not available in traffic data set, so we assume the distance between all sensor node pairs are equal for DESM and STI.

Figure 7(d), 7(e), 7(f), 7(g) and 7(f) show the result of random split. Only the RMSEs of the most competitive models are shown and those with higher RMSEs are cut out.

Let us focus on random split first. In random split, we can see that MF perform very badly. However, after adding the temporal regularization, TR-MF shows significant improvement and outperforms others greatly in all cases. On the other hand, linear interpolation is quite competitive on berkeley data, especially when the size of training set is large. This

is because the sampling rate of Berkeley data set is fairly high, so the temporal correlation is prominent. However, the performance of linear interpolation deteriorates a lot in traffic data set, which corresponds to the low sampling rate.

5.2.1 Temporal Split

Table 2, 3 and 4 show the result of temporal split on Berkeley data set and Table 5 and 6 show the result of temporal split on traffic data set. In the temporal split, it is clear that linear interpolation performs terribly since its prediction is simply the last reading of the sensor. Our TR-MF and AKE show the best performance in temporal split.

Table 2. RMSE of (Berkeley, temporal, humidity)

train	LI	TR-MF	AKE	DESM	STI	MI
10%	$4.183_{(4)}$	$0.957_{(2)}$	$1.669_{(3)}$	$4.185_{(5)}$	$\mathbf{0.792}_{(1)}$	2.710
20%	$5.421_{(4)}$	$\mathbf{0.796}_{(1)}$	$0.969_{(3)}$	$5.422_{(5)}$	$0.865_{(2)}$	2.193
40%	$6.443_{(4)}$	$\mathbf{0.771}_{(1)}$	$1.025_{(3)}$	$6.443_{(5)}$	$0.961_{(2)}$	1.629
60%	$2.223_{(4)}$	$\mathbf{0.540}_{(1)}$	$0.928_{(3)}$	$2.233_{(5)}$	$0.864_{(2)}$	1.284
80%	$1.208_{(4)}$	$\mathbf{0.447}_{(1)}$	$0.636_{(2)}$	$1.216_{(5)}$	$0.688_{(3)}$	1.171
85%	$2.306_{(4)}$	$\mathbf{0.323}_{(1)}$	$0.777_{(2)}$	$2.308_{(5)}$	$0.796_{(3)}$	0.966
rank	4.00	1.17	2.67	5.00	2.17	

Table 3. RMSE of (Berkeley, temporal, light)

train	LI	TR-MF	AKE	DESM	STI	MI
10%	$320.0_{(5)}$	$\mathbf{220.1}_{(1)}$	$239.1_{(2)}$	$320.0_{(4)}$	$251.3_{(3)}$	331.2
20%	$497.8_{(4)}$	$\mathbf{113.3}_{(1)}$	$257.1_{(3)}$	$499.7_{(5)}$	$206.6_{(2)}$	412.7
40%	$194.5_{(3)}$	$\mathbf{58.1}_{(1)}$	$68.5_{(2)}$	$195.2_{(4)}$	$208.3_{(5)}$	99.6
60%	$312.1_{(4)}$	$\mathbf{41.7}_{(1)}$	$77.5_{(2)}$	$312.1_{(5)}$	$289.2_{(3)}$	153.6
80%	$293.5_{(4)}$	$\mathbf{21.4}_{(1)}$	$84.9_{(2)}$	$293.9_{(5)}$	$213.9_{(3)}$	201.3
85%	$277.9_{(4)}$	$\mathbf{8.3}_{(1)}$	$79.0_{(2)}$	$280.4_{(5)}$	$92.7_{(3)}$	198.4
rank	4.00	1.00	2.17	4.67	3.17	

Table 4. RMSE of (Berkeley, temporal, temperature)

10%	$3.760_{(4)}$	$0.515_{(3)}$	$0.514_{(2)}$	$3.761_{(5)}$	$\mathbf{0.492}_{(1)}$	1.712
20%	$2.320_{(4)}$	$\mathbf{0.392}_{(1)}$	$0.406_{(2)}$	$2.321_{(5)}$	$0.415_{(3)}$	1.985
40%	$3.595_{(4)}$	$0.310_{(2)}$	$\mathbf{0.304}_{(1)}$	$3.595_{(5)}$	$0.486_{(3)}$	1.788
60%	$1.960_{(4)}$	$\mathbf{0.206}_{(1)}$	$0.340_{(2)}$	$1.962_{(5)}$	$0.532_{(3)}$	1.541
80%	$0.887_{(4)}$	$\mathbf{0.132}_{(1)}$	$0.280_{(2)}$	$0.894_{(5)}$	$0.326_{(3)}$	0.971
85%	$1.024_{(5)}$	$\mathbf{0.088}_{(1)}$	$0.305_{(2)}$	$1.016_{(4)}$	$0.351_{(3)}$	0.644
rank	4.17	1.50	1.83	4.83	2.67	

5.3 Results on STR-MF

Here we focus on Berkeley data set because only in which the location information of sensor nodes is available.

Table 7 and 8 compare STF-MF with TR-MF. STR-MF mean TR-MF with strong spatial regularization, while sTR-MF is with weak spatial regularization. We got no improvement in all random split data and either (temporal, light). However, we see significant improvement on (temporal, humidity) and (temporal, temperature).

There are two factors that decide the successfulness of spatial regularization. On the one hand, TR-MF already captures some of the spatial correlation. Since we have thousands of readings for every sensor node, TR-MF can learn the spatial correlation from the data and create similar factors

Table 5. RMSE of (traffic, temporal, humidity)

train	LI	TR-MF	AKE	DESM	STI	MI
10%	$27.751_{(4)}$	$5.195_{(2)}$	$\mathbf{5.081}_{(1)}$	$27.794_{(5)}$	$5.752_{(3)}$	5.920
20%	$21.469_{(4)}$	$5.487_{(2)}$	$\mathbf{5.020}_{(1)}$	$22.372_{(5)}$	$5.641_{(3)}$	5.504
40%	$25.828_{(4)}$	$5.782_{(2)}$	$\mathbf{5.193}_{(1)}$	$25.997_{(5)}$	$5.957_{(3)}$	5.905
60%	$27.489_{(4)}$	$\mathbf{4.954}_{(1)}$	$5.027_{(2)}$	$27.517_{(5)}$	$6.372_{(3)}$	5.457
80%	$18.936_{(4)}$	$4.564_{(2)}$	$\mathbf{4.194}_{(1)}$	$19.398_{(5)}$	$4.972_{(3)}$	4.789
85%	$23.513_{(4)}$	$4.248_{(2)}$	$\mathbf{3.666}_{(1)}$	$23.615_{(5)}$	$4.695_{(3)}$	5.064
rank	4.00	1.83	1.17	5.00	3.00	

Table 6. RMSE of (traffic, temporal, temperture)

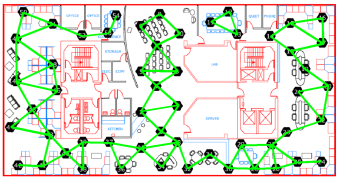
train	LI	TR-MF	AKE	DESM	STI	MI
10%	$11.435_{(4)}$	$1.700_{(2)}$	$\mathbf{1.679}_{(1)}$	$11.551_{(5)}$	$2.089_{(3)}$	1.785
20%	$7.186_{(4)}$	$1.812_{(2)}$	$\mathbf{1.664}_{(1)}$	$7.259_{(5)}$	$1.956_{(3)}$	1.765
40%	$10.594_{(4)}$	$1.832_{(2)}$	$\mathbf{1.579}_{(1)}$	$10.639_{(5)}$	$2.048_{(3)}$	1.724
60%	$14.134_{(4)}$	$\mathbf{1.666}_{(1)}$	$1.687_{(2)}$	$14.251_{(5)}$	$2.780_{(3)}$	1.712
80%	$16.166_{(5)}$	$1.430_{(2)}$	$\mathbf{1.414}_{(1)}$	$16.139_{(4)}$	$1.710_{(3)}$	1.594
85%	$10.022_{(5)}$	$1.441_{(2)}$	$\mathbf{1.356}_{(1)}$	$9.992_{(4)}$	$1.719_{(3)}$	1.389
rank	4.33	1.83	1.17	4.67	3.00	

Table 7. RMSE of Berkeley Random split

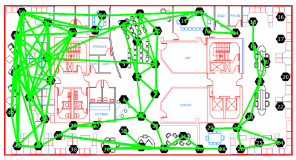
	Humidity			Light			Temperature		
train	TR-MF	STR-MF	sTR-MF	TR-MF	STR-MF	sTR-MF	TR-MF	STR-MF	sTR-MF
10%	**0.142**	0.484	0.173	**35.5**	97.4	38.3	**0.046**	0.154	0.061
20%	**0.114**	0.424	0.135	**28.2**	90.6	28.9	**0.032**	0.146	0.047
40%	**0.092**	0.352	0.104	**21.2**	85.8	22.8	**0.023**	0.145	0.037
60%	**0.082**	0.337	0.093	**17.2**	83.3	18.3	**0.018**	0.147	0.031
80%	**0.076**	0.324	0.084	**17.7**	84.4	18.1	**0.015**	0.148	0.027
85%	**0.075**	0.326	0.083	**14.4**	82.0	15.8	**0.016**	0.138	0.028

Table 8. RMSE of Berkeley Temporal split

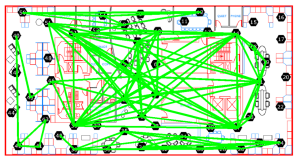
	Humidity			Light			Temperature		
train	TR-MF	STR-MF	sTR-MF	TR-MF	STR-MF	sTR-MF	TR-MF	STR-MF	sTR-MF
10%	0.957	0.573	**0.547**	**220.070**	281.607	264.763	0.515	**0.242**	0.307
20%	0.796	0.657	**0.459**	**113.310**	236.865	230.054	0.392	**0.179**	0.187
40%	0.771	0.520	**0.455**	**58.090**	110.758	64.388	0.310	0.196	**0.189**
60%	0.540	**0.351**	0.708	**41.730**	150.149	69.235	0.206	**0.191**	0.243
80%	0.447	0.299	**0.261**	**21.450**	112.694	28.017	0.132	**0.108**	0.114
85%	0.323	**0.166**	0.256	**8.310**	85.423	12.079	0.088	**0.065**	0.082



(a) Expected Similarity



(b) Humidity: top 100 similar pairs



(c) Light: top 100 similar pairs

for similar sensor nodes. On the other hand, the neighboring pairs in the deployment graph may not imply the actual similarity. For example, we can look at the similarity graph of humidity and light sensors. We define the similarity of sensor node n_i and n_j as by the factor similarity: $\frac{q_{n_i}^T q_{n_j}}{max(||q_{n_i}||^2, ||q_{n_j}||^2)}$, and we get the similarity graph:

We can see that the similarity graph of humidity is more similar to our expectation, while the similarity graph of light is not. We can only obtain improvement when the data information is insufficient, which is the temporal split case, and when our expectation similarity is not far from the actual similarity, which is the temperature and humidity.

5.4 Multivariate Learning

5.4.1 Multivariate TR-MF

To investigate the effectiveness of multivariate learning, temporal and humidity are selected, and we want to know whether one can help with the prediction of the other. Here we compare two models: MTR-MF and TF, which consider the multivariate correlation, with the univariate model TF-MF.

We have two settings in this experiment. Assume we are predicting the missing entries in the temperature matrix. In the first setting (denoted as TR-MF-all), we include all humidity entries have (i.e. training, validation and testing set), and see whether this helps with the prediction. In the second (denoted as TR-MF-part), we include only the observed entries of humidity matrix as those in temperature matrix. However, to predict an entry in temperature matrix, TF needs the corresponding enty in the humidity matrix. Thus, we will first predict that humidity entry by the humidity information by TR-MF and use it to do TF.

In general, we see that TR-MF-all is better then TR-MF, while TR-MF-part is about the same as TR-MF.

On the other hand, TF works well on traffic dataset while it does't take advantage from multi-source signal on Berkeley dataset. Sensors from traffic dataset are deployed outdoor so the variance of heterogeneous data is large. While sensors from Berkeley dataset are deployed indoor, it cause the variance of heterogeneous data is small. The small variance makes TF does't work well because we quantize the heterogeneous signal into nominal bins before training. Most of heterogeneous sensor readings are divided into the same bin such that the additional dimension of TF is redundant. We finally make a conclusion that the high variance heterogeneous data are more suitable to use TF model.

don't use result here!!! see my google doc!

Table ? and ? show the TF result of random split and temporal split on Berkeley dataset while table ? and ? show the TF result of random split and temporal split on traffic dataset.

5.5 Prediction Performance

In this section, we measure the performance of missing value estimation algorithms by regression methods. We demonstrate the experiment on traffic dataset. The readings of gateway node are to be predicted in offline mode. For a given time step, the data from gateway serves as the label of a instance while the data from the other sensor nodes serve as features. We designate 80% of the instances for training data, while the remaining part is used as testing data. We first fill the features from training and testing data by their global means, linear interpolation, KNN, and TR-MF model. The regression models we choose are linear regression(LR) and support vector regression(SVR), these being linear and nonlinear, respectively. Table ? show the results with different data missing rates. It is obvious that a more accurate fill value will help the regression model in predicting the objective sensor more precisely.

Table 9. predict gateway humidity by LR (RMSE)

	Filling method			
missing rate	global mean	LI	Hybrid-KNN	TR-MF
5%	4.144	3.857	2.512	2.473
10%	4.143	3.868	2.597	2.526
30%	5.161	3.955	2.773	2.470
50%	6.234	4.282	3.158	2.756
70%	7.728	5.082	3.310	2.743
80%	9.019	7.458	3.306	2.802

Table 10. predict gateway humidity by SVR (RMSE)

	Filling method			
missing rate	global mean	LI	Hybrid-KNN	TR-MF
5%	4.252	3.980	2.609	2.567
10%	3.933	4.006	2.749	2.591
30%	5.172	4.083	2.814	2.532
50%	6.234	4.384	3.260	2.813
70%	7.686	5.235	3.313	2.822
80%	9.039	8.508	3.464	2.910

6 Discussion

6.1 Parameter Setting

For both MF and TF models, we have realized that the parameter setting in random split has to be different from that in temporal split. In a nutshell, wstronger regularization is needed in temporal split. Below we provide some qualtative analysis using examples.

Equation xxx shows a random split case and xxx shows a temporal split case. In random split, the confidence for prediction is usually higher and the model relies mostly on temporal correlation, therefore it is preferred to use more factors

and weaken the strength of regularization. We also see that linear interpolation can be viewed as a solution of TRMF, in which the number of factors is large and no conventional regularization is applied.

$$\begin{bmatrix} 16 & 22 & 18 \\ ? & 22 & 19 \\ 18 & ? & 22 \\ 19 & 24 & ? \\ 20 & 26 & 26 \end{bmatrix} = \begin{bmatrix} 16 & 22 & 18 \\ \mathbf{17} & 22 & 19 \\ 18 & \mathbf{23} & 22 \\ 19 & 24 & \mathbf{24} \\ 20 & 26 & 26 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.1)$$

Temporal split is like Equation 6.2. In contrast to temporal split case, we can't use temporal correlation to infer the missing values and we have less confidence in prediction. We can only rely on the spatial correlation, so we prefer fewer factors and stronger conventioal regularization.

$$\begin{bmatrix} 30 & 30 & 28 \\ 31 & 31 & 28 \\ 32 & ? & 28 \\ 33 & ? & 28 \\ 32 & ? & 28 \end{bmatrix} = \begin{bmatrix} 30 & 28 \\ 31 & 28 \\ 32 & 28 \\ 33 & 28 \\ 32 & 28 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.2)$$

To be specific, we share the actual values of the parameters. The learning rate η is 0.04 to 0.004 for MF and 0.001 to 0.0001 for TF, which is chosen as the largest value that still maintains stable optimization process. A strong temporal regularization is always preferred. Temporal regularization times learning rate ($\gamma \times \eta$) is set to 0.2 in all of our experiment. The conventional regularization is 0.001 for temporal split and 0 for random split in MF, while in TF, it is 0.005 for temporal split and 0.001 for random split. Finally, we observe that a large K (although slower) doesn't lead to overfitting because SGD is a natural regularization that prefers small factor values. The number of factors K is 54 for MF and 30 for TF in Berkeley data set, and it is 21 for MF and 10 for TF in traffic data set.

6.2 Time Complexity

Suppose the number of factors is K. For MF (TR-MF and STR-MF), the number of time steps is F_1 and the number of sensor nodes is F_2. For TF, the numbers of three features are F_1, F_2, F_3. If there are R regularization terms (spatial and temporal), and there are N observed entries from the training set, then the time spent in each update is $\Theta(KN+KR)$ (independent of F_1, F_2 and F_3)for both MF and TF. The total iteration number varies with data and its missing rate. However, the training time is not the main concern in our application, since we only need to train our model once.

The imputation time is more important as we need the values for further learning task like event classification. Both MF and TF are very efficient in the testing phase. To predict a missing value, it takes $2K$ multiplication operations to MF and $3K$ multiplication operations to TF. Since K is very small so the prediction can be done in real time. On a normal laptop, it takes less than a second to predict all values for our Berkey data matrix (270000 values).

7 Conclusion

Conclusion here!!

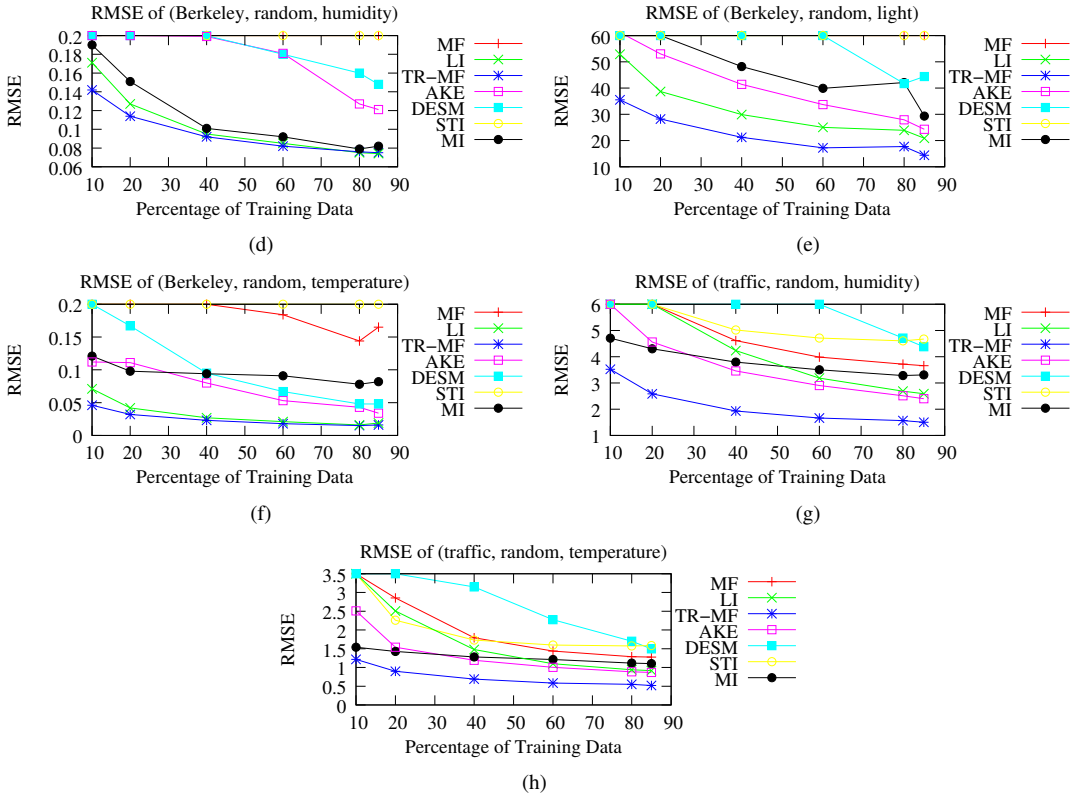


Figure 7. Random Split

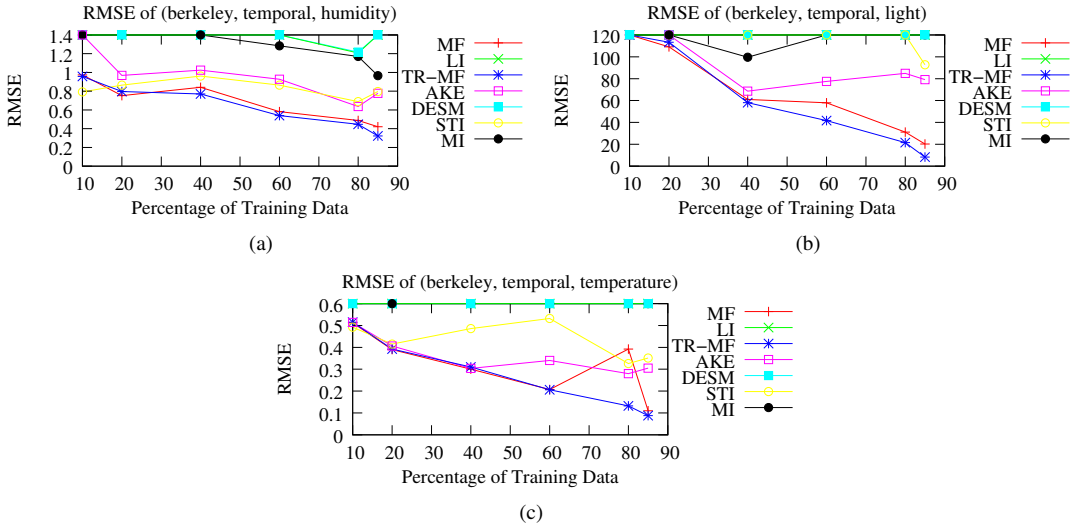


Figure 8. Temporal Split

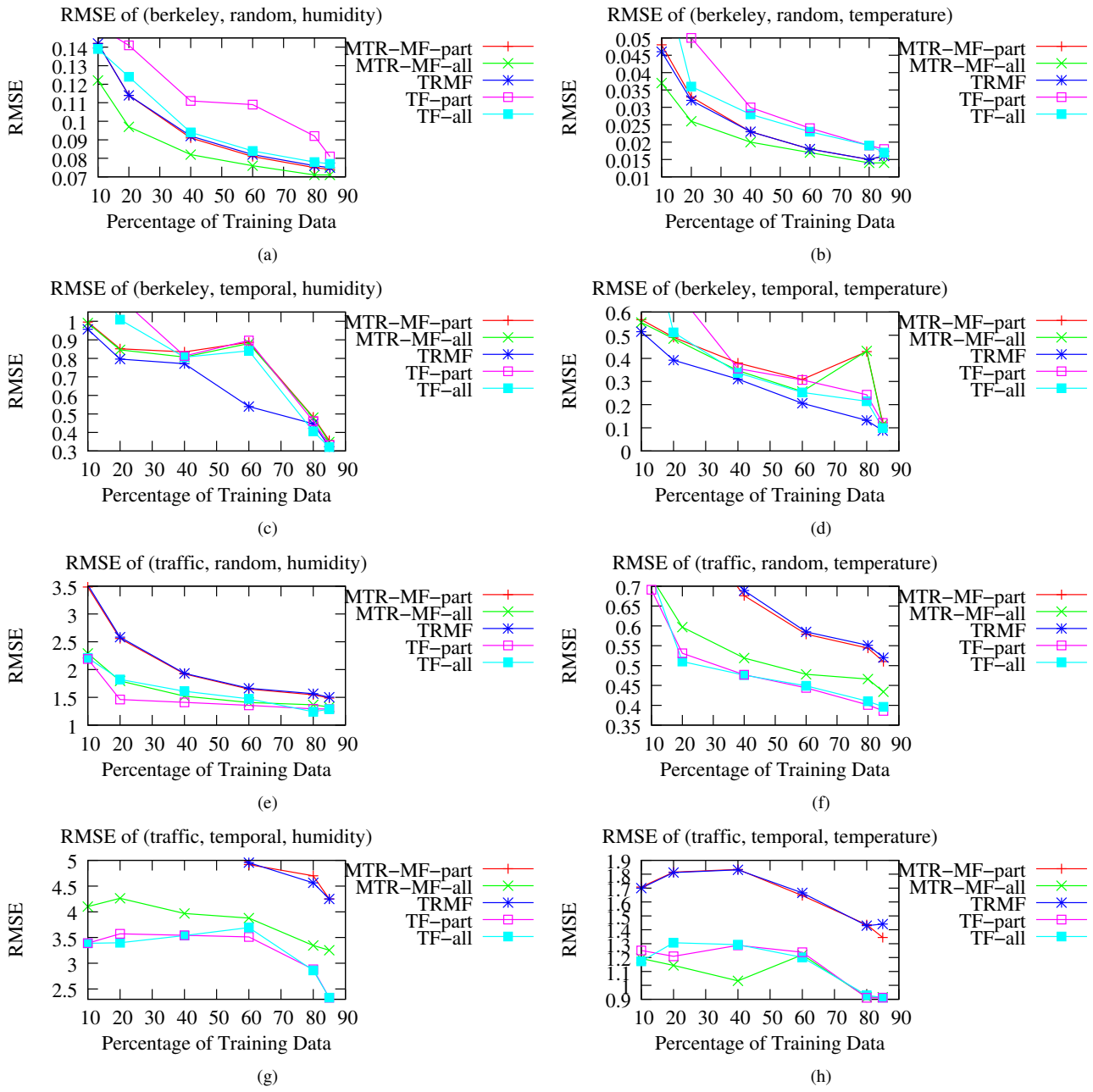


Figure 9. the performance of Multivariate models