Distributed Multi-task Learning for Sensor Network

Jiyi Li[1], Tomohiro Arai[1], Yukino Baba[1], Hisashi Kashima[1], Shotaro Miwa[2]

[1] Department of Intelligence Science and Technology, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, Japan, 606-8501
{jyli,baba,kashima}@i.kyoto-u.ac.jp, arai@ml.ist.i.kyoto-u.ac.jp
[2] Mitsubishi Electric Corporation
Miwa.Shotaro@bc.mitsubishielectric.co.jp

Abstract. A sensor in a sensor network is expected to be able to make prediction or decision utilizing the models learned from the data observed on this sensor. However, in the early stage of using a sensor, there may be not a lot of data available to train the model for this sensor. A solution is to leverage the observation data from other sensors which have similar conditions and models with the given sensor. We thus propose a novel distributed multi-task learning approach which incorporates neighborhood relations among sensors to learn multiple models simultaneously in which each sensor corresponds to one task. It may be not cheap for each sensor to transfer the observation data from other sensors; broadcasting the observation data of a sensor in the entire network is not satisfied for the reason of privacy protection; each sensor is expected to make real-time prediction independently from neighbor sensors. Therefore, this approach shares the model parameters as regularization terms in the objective function by assuming that neighbor sensors have similar model parameters. We conduct the experiments on two real datasets by predicting the temperature with the regression. They verify that our approach is effective, especially when the bias of an independent model which does not utilize the data from other sensors is high such as when there is not plenty of training data available.

Keywords: Sensor Network, Multi-task Learning, Distributed Approach

1 Introduction

Traditional data integration and analysis for sensor data collects the observation data from all sensors to construct *centralized* global models at a data center (Figure 1.(a)). With the advances in the device technology, the computing power and storage capacity of sensors have been improved. Meanwhile, with the large scale of sensor network growth, transferring all observation data in the network and learning the centralized model utilizing them require a huge amount of computation. Therefore, a sensor in a sensor network is expected to be able

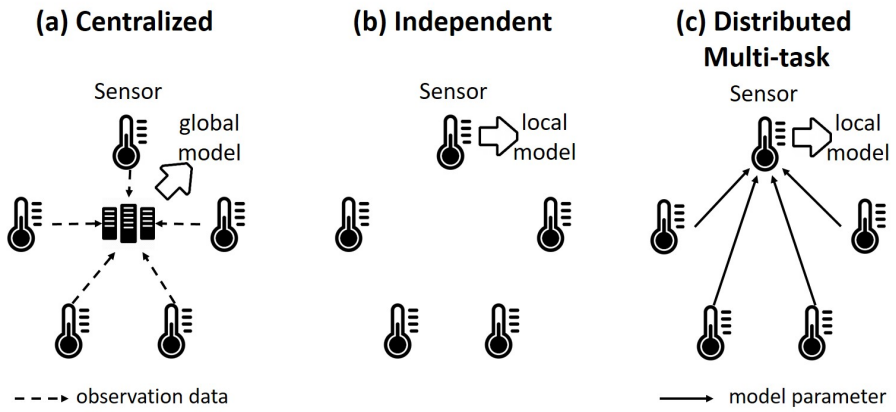


Fig. 1. Different Types of Solutions for Learning the Models of Sensors in Sensor Network: (a). Centralized Global Model, (b). Independent Local Model, (c). Distributed Multi-task Local Model (Our Proposal).

to make prediction or decision by itself utilizing the decentralized local models learned from the data observed on this sensor.

A naïve solution of learning a local model by a sensor is only utilizing the observation data on this sensor. We define it as *independent* model (Figure 1.(b)). One of the issues that need to be solved for learning such local models is the limitation of available data for training. In the early stage of using a sensor, there may be not a lot of data available to train the local model for this sensor. A solution is to leverage the observation data from other sensors which have similar conditions and thus may have similar models with the given sensor. However, sensors can be overloaded to transfer the observation data to other sensors, and broadcasting the observation data of each sensor in the entire network has the privacy leakage problem. In addition, in the prediction stage, we hope each sensor is able to make prediction independently from the observation data from neighbor sensors.

To overcome these problems, based on the assumption that neighbor sensors have similar model parameters, we propose sharing the model parameters among the sensors instead of transferring the observation data. We treat the problems of learning the local models for a set of sensors as a *multitask* learning problem in which each sensor corresponds to one task. In the learning process, the proposed method incorporates the neighborhood relations among sensors to learn multiple models simultaneously. We then present a *distributed* learning algorithm for our multi-task model. In this algorithm, at each iteration, the local model of each sensor is computed simultaneously, and then the model parameters are shared among all the sensors (Figure 1.(c)). The parameters are incorporated in the regularization terms of the objective function of each sensor, thus we can use the information from the other sensors to update each local model. Although

a few algorithms for distributed multi-task learning algorithms for client-server models have been proposed [6, 17], we propose a novel distributed algorithm for multi-task learning in peer-to-peer models.

In this paper, we focus on the time series regression tasks to verify our approach. We utilize several real datasets collected from real sensor networks in which the sensors recorded the temperature information for different applications. The task is to predict the temperature at a given place with a sensor based on the historical temperature observation data. The experiments show that our approach is effective, especially when the volume of training data is small which is one of the reasons of the under-fitting of an independent model learned based on the data on a sensor.

Furthermore, although a centralized model generally is expected to have better performance than local models because it can utilize much more observation data for learning, in some cases, because the environmental conditions and data characteristics of the sensors may be somewhat different, a local model for a sensor may be able to provide better presentation of this sensor when the training data is not a lot.

The contributions of this paper are as follows.

- We focus on the problem of predictive modeling in sensor networks in which each sensor learns a local model by utilizing its own data and the data from neighbor sensors. We propose a distributed approach which the sensors learn multiple models simultaneously.
- We propose a novel approach to solve the problem in a manner of multi-task learning by treating each sensor as a task. To avoid transferring observation data of neighbor sensors in the network, our approach only leverages the model parameters from neighbor sensors.
- We use real datasets to verify that our approach is effective especially when the bias of an independent model which does not utilize the data from other sensors is high such as when there is not plenty of training data available.

2 Related Work

We review the existing work on the following four issues, i.e., multi-task learning, distributed learning, prediction of sensor data and time-series data mining.

Multi-Task Learning: Rather than implementing multiple related learning tasks separately, multi-task learning leverages the information from different tasks to train these tasks simultaneously so that it can raise the model performance of each single task [4]. In this paper, because learning the model of each sensor can be regarded as one learning task, and modeling all sensors can be simultaneously performed. We thus solve our problem with multi-task learning.

Existing work on multi-task learning models the relatedness and model parameters of multiple tasks in different manners. Argyriou et al. [2] assumed the common underlying representation shared across multiple related tasks and

learned a low-dimensional representation. Evgeniou et al. [8] modeled the relation between tasks in terms of a kernel function that used a task coupling parameter. Evgeniou et al. [7] also leveraged the similarity between tasks as prior knowledge, assumed that similar tasks have similar parameters and utilized as regularization. In our work, we model the task relatedness and model parameters of multiple tasks by using the manner proposed in [7]. In our scenario, the similarity between tasks can be given by the neighborhood graph based on the physical distance between sensors. In other words, the sensors which are near to each other are regarded to have similar parameters, and we utilize this similarity to perform regularization.

Regarding the implementation of multi-task learning in a distributed environment, Wang et al. [17] proposed distributed learning method followed the assumption of the common underlying representation like [2]. The manner of the common underlying representation requires the distributed learning method having a client-server manner. Our approach and [17] differ in the regularization schemes. [17] incorporated shared-sparsity among local models, while ours makes local model parameters close to each other. Dinuzzo et al. [6] also proposed another method of distributed multi-task learning with a client-server manner. However, the client-server manner is not suitable for our scenario in which a server does not exist. In contrast, in our distributed approach, the tasks learn simultaneously in a peer-to-peer manner. In addition, Vanhaesebrouck et al. [16] proposed asynchronous algorithms for collaborative peer-to-peer network based on label propagation which jointly learned and propagated their model based on both their local dataset and the behavior of their neighbors.

Distributed Learning: The idea of conducting learning in a distributed manner has also been used for various areas related to machine learning and data mining. For example, in the area of reinforcement learning, multi-agent learning [14] performs learning of agents in distributed cooperative manner. [12] proposed parallel and distributed learning approach in which a number of processors cooperated together for a single learning task. In contrast, our work concentrates on the area of sensor data and multiple tasks.

Prediction of Sensor Data: With the development of technology such as Internet of Things, sensors have reach miniaturization, high performance, and low price. The sensor networks in which the sensors can communicate to each other and have some computation capability have become possible. In this research, we focus on the prediction tasks for sensor observation data. There are several previous studies on time series modeling of sensor data, e.g., [15], while they do not deal with collaborative learning among sensors. [10] is the existing work which is most similar to our work. It proposed distributed regression model based on divided local region. However, it needs the observation data from neighbor sensors in the prediction stage. In addition, Ahmadi et al. [1] proposed a solution with a centralized clustering on entire area and an independent method for each cluster. Ceci et al. [5] used neighbor distances as features for a centralized model. In

contrast, in our approach, the predictions of a sensor can be made independently from the observation data of neighbor sensors.

Time Series Data Mining: There is much existing work on learning the prediction model from time series data. The regression model is just one of the alternatives which may be not the best model in the existing work. For example, the approaches based on RNN with LSTM [11, 9, 13] may have better prediction performance on non-stationary time series data. Our work does not exactly focus on the prediction task for time series data. We focus on the distributed multi-task learning in a sensor network by utilizing the regression task to illustrate our idea. Although the sensor datasets we use are time series data, our proposal can actually also be utilized to the other kinds of regression tasks.

3 Our Approach

We describe our approach in details in this section. Our purpose is to learn decentralized local models for different sensors in the network. The learning process thus needs to be carried out at each corresponding sensor, instead of a data center which collects all observation data from the entire sensor network. We propose a distributed multi-task learning approach in which each sensor corresponds to one task to solve this problem.

The type of learning task on a sensor can be diverse based on the practical requirements. In this paper, we focus on time series regression tasks, in which we are given observed data in a specific time window and aim to predict the value of the observations in future timestamps. Such time-series prediction tasks are very useful and widely required for sensor networks, e.g., predicting the temperature at a place with a weather sensor. Our proposal is not limited to such specific prediction task. Our approach can be directly utilized to other kinds of regression tasks. Our idea of distributed multi-task learning can also be adapted to other types of learning tasks such as classification.

3.1 Preliminary

For a given sensor network $\mathcal{S}$ ($|\mathcal{S}| = n$) which generates observation data based on time series, we learn a local regression model f_k for predicting the value of observation data at timestamp i on each sensor $s_k \in \mathcal{S}$ based on the observation data at a time window before i with a fixed window size t. k is the index of a sensor. The feature data on a sensor s_k is $\mathcal{X}_k = (\mathbf{x}_{k1}, \mathbf{x}_{k2}, \ldots, \mathbf{x}_{kn_k})^\top$ which contains both the observation data in a specific time window on this sensor and an intercept term with a value of 1, $\mathbf{x}_{ki} = (x_{k(i-t)}, \ldots, x_{k(i-2)}, x_{k(i-1)}, 1)$. n_k is the number of instances on sensore s_k. We denote the target value as $\mathbf{y}_k = (y_{k1}, y_{k2}, \ldots, y_{kn_k})^\top$ in which $y_{ki} = x_{ki}$ is the predicted observation value at timestamp i.

Given feature data $\{\mathcal{X}_k\}_k$ and target data $\{\mathbf{y}_k\}_k$ from a sensor network $\mathcal{S} = \{s_k\}_k$, our task is to learn a local regression model for each sensor s_k with model parameters $\mathbf{w}_k$.

The basic linear regression model f_k and the object function $\mathcal{L}_0$ which are used for the independent approach in this paper is as follows. Because we focus on our idea of the distributed multi-task learning, we thus utilize this fundamental regression model without any additional settings such as regularization to highlight the effectiveness of our proposal.

$$f_k(\mathcal{X}_k, \mathbf{w}_k) = \mathbf{w}_k^\top \mathcal{X}_k, \quad \mathcal{L}_0(\mathbf{w}_k) = ||\mathbf{w}_k^\top \mathcal{X}_k - \mathbf{y}_k||_2^2, \quad \mathbf{w}_k = (\mathcal{X}_k^\top \mathcal{X}_k)^{-1} \mathcal{X}_k^\top \mathbf{y}_k. \quad (1)$$

3.2 Multi-Task Model

To learn the local model on a given sensor, one of the problems that need to be solved is the limited available data for training. In the early stage of using a sensor, there may be not a lot of data available to train the local model for this sensor. A solution is to leverage the observation data from other sensors. The other sensors in the sensor network may have similar but somewhat different models with the given sensor. In the learning process, all these sensors can communicate with each other to learn multiple models simultaneously. The problem of learning the local models for a set of sensors can thus be regarded as a multi-task learning problem in which each sensor corresponds to one task. We thus propose a multi-task learning approach for learning the decentralized local models of the sensors.

In the training stage, sensors are allowed to transfer the data with other sensors. In a sensor network, there may be huge amount of sensors installed in an immense geographical range. For a given sensor, it costs too much to transfer the data from all other sensors. It would be better to constrain that each sensor can only communicate with the neighbor sensors which are physically near to the given sensor. Considering that the neighbor sensors are more possible to have more similar conditions with the given sensor because of shorter physically distance, this settings of using neighbor sensors only is rational and cost-effective. We denote the set of neighbor sensors of a given sensor s_k as $\mathcal{S}_k$.

Which kinds of data are transferred with neighbor sensors and used for learning models is another issues needs to be solved. Intuitively, we can transfer and use the observation data. However, there are three main problems in this manner. First, it may be not cheap for each sensor to keep transferring the observation data with a number of other sensors; Second, broadcasting the observation data of an sensor in the entire network have the privacy leakage problem; Third, each sensor needs to be able to make computation in the prediction stage without waiting for using the observation data from neighbor sensors. To overcome these problems, instead of transferring the observation data in the network, our distributed approach in the network transfers the model parameters with neighbor sensors. This solution is based on the assumption that neighbor sensors have similar model parameters. To leverage the model parameters from neighbor sensors, our approach shares them as the regularization terms in the objective function. Note that all sensors are not necessary to have the same model.

Based on the above discussion, we propose a novel distributed multi-task learning approach which incorporates neighborhood relations among sensors to learn multiple models simultaneously in which each sensor corresponds to one task. Following [7], the overall object function of all local regression models for all sensors can be written as follows.

$$\mathcal{L}(\{\mathbf{w}_k\}_k) = \sum_{k=1}^{n} ||\mathbf{w}_k^\top \mathcal{X}_k - \mathbf{y}_k||_2^2 + \lambda \sum_{k=1}^{n} \sum_{s^{k'} \in \mathcal{S}_k} ||\mathbf{w}_k - \mathbf{w}_{k'}||_2^2 \tag{2}$$

λ is the hyperparameter to control the regularization term in the object function. In this paper, we use same λ to all neighbor sensors. An option is to set or learn different λ for a different pair of sensors. For the purpose of clarifying our proposal of distributed multi-task learning of our problem settings, we use this simpler solution with single λ hyperparameter.

The object function $\mathcal{L}_k(\mathbf{w}_k)$ of a local regression model for a sensor $\mathbf{s}_k$ can thus be written as follows.

$$\mathcal{L}_k(\mathbf{w}_k) = ||\mathbf{w}_k^\top \mathcal{X}_k - \mathbf{y}_k||_2^2 + \lambda \sum_{s_{k'} \in \mathcal{S}_k} ||\mathbf{w}_k - \mathbf{w}_{k'}||_2^2 \tag{3}$$

We can utilize formulation (3) for each sensor respectively instead of utilizing formulation (2) for all sensors. Because this formulation (3) for the original formulation (2) can be regarded as a block-coordinate descent for a convex function, the convergence of the formulation (3) can be guaranteed [3].

The parameters $\mathbf{w}_k$ in our multi-task model which minimizes this object functions can be computed as follows, in which m_k is the size of $\mathcal{S}_k$.

$$\mathbf{w}_k = (\mathcal{X}_k^\top \mathcal{X}_k + m_k \lambda \mathcal{I})^{-1} (\mathcal{X}_k^\top \mathbf{y}_k + \lambda \sum_{s_{k'} \in \mathcal{S}_k} \mathbf{w}_{k'}) \tag{4}$$

3.3 Distributed Learning

This multi-task formulation in the previous sub-section shows that it requires integrating the $\mathbf{w}_k$ for the given sensors $\mathbf{s}_k$ and its neighbor sensors $\mathcal{S}_k$ to compute together for solving the optimal values of $\{\mathbf{w}_k\}_k$. Generally, it can be implemented at a data center which collects the observation data from all sensors. However, based on our scenario discussed in this paper, we need to distribute the computation to each sensor. We thus propose a distributed approach for our multi-task model.

In our distributed approach, the local model parameters of the sensors are computed on multiple sensors simultaneously. On a given sensor $\mathbf{s}_k$, in each iteration r, the model parameter $\mathbf{w}_k^{(r)}$ is updated based on the values of model parameters $\mathbf{w}_{k'}^{(r-1)}$ of other neighbor sensors $s_{k'} \in \mathcal{S}_k$ at iteration $r-1$. The formulation of the multi-task model can be revised in the following manner.

$$\mathbf{w}_k^{(r)} = (\mathcal{X}_k^\top \mathcal{X}_k + m_k \lambda \mathcal{I})^{-1} (\mathcal{X}_k^\top \mathbf{y}_k + \lambda \sum_{s_{k'} \in \mathcal{S}_k} \mathbf{w}_{k'}^{(r-1)}) \tag{5}$$

Algorithm 1: Distributed Multi-Task Learning

Input: Feature Data $\{\mathcal{X}_k\}_k$; Target Data $\{\mathbf{y}_k\}_k$
Output: Model Parameters $\{\mathbf{w}_k\}_k$;
// Initialization
forall $s_k \in \mathcal{S}$ **do**
  $\mathbf{w}_k^{(0)} = (\mathcal{X}_k^\top \mathcal{X}_k)^{-1} \mathcal{X}_k^\top \mathbf{y}_k$;
end
$r = 1$;
// Learning
forall $s_k \in \mathcal{S}$ **do**
  while $r \leq r_{max}$ **do**
    // broadcast the parameters to neighbor sensors
    forall $s_{k'}$, $s_k \in \mathcal{S}_{k'}$ **do**
      Send($\mathbf{w}_k^{(r-1)}$);
    end
    // collect the parameters from neighbor sensors
    forall $s_{k'} \in \mathcal{S}_k$ **do**
      Receive($\mathbf{w}_{k'}^{(r-1)}$);
    end
    // update parameters at iteration r based on the parameters of neighbor sensors at iteration r-1
    $\mathbf{w}_k^{(r)} = (\mathcal{X}_k^\top \mathcal{X}_k + m_k \lambda \mathcal{I})^{-1} (\mathcal{X}_k^\top \mathbf{y}_k + \lambda \sum_{s_{k'} \in \mathcal{S}_k} \mathbf{w}_{k'}^{(r-1)})$;
    $r = r + 1$;
    if $||\mathbf{w}_k^{(r)} - \mathbf{w}_k^{(r-1)}||_2^2 < \theta$ **then**
      break;
    end
  end
end
return $\{\mathbf{w}_k\}_k$

In the distributed computation, a sensor $\mathbf{s}_k$ sends $\mathbf{w}_k^{(r-1)}$ to other neighbor sensors $s_{k'}$ with $s_k \in \mathcal{S}_{k'}$, and waits for receiving the current model parameters at iteration $r-1$ from the neighbor sensors $s_k \in \mathcal{S}_k$. After that, it updates the model parameters $\mathbf{w}_k^{(r)}$ at iteration r based on its own observation data ($\mathcal{X}_k$, $\mathbf{y}_k$) and the model parameters from neighbor sensors. When all updated model parameters from neighbor sensors in $\mathcal{S}_k$ have reached the sensor $\mathbf{s}_k$, it starts the iteration $r+1$ of computation. There are two stop criteria of the iterations, i.e,. the difference of model parameters between two continue iterations are smaller than a threshold θ or the maximum number of iterations has been reached. Algorithm 1 lists the computation of our distributed approach.

4 Experiments

In this section, we verify our approach with data collected from real sensor networks. All these data are time-series temperature observation data, while the

sensor networks are from different applications. We utilize our approach to solve the problem of temperature prediction of a given place with a sensor in a sensor network at a timestamp.

4.1 Dataset: Weather

The weather observation data such as temperature, precipitation, sunshine duration, wind and so on are collected by the sensors installed at specific places by the national meteorological agency. These collected data then can be used for various purposes of data analysis such as predicting the weather in the future, analyzing weather conditions in the past and so on. These sensors construct a huge weather sensor network around a country.

In this paper, we collect the weather observation data from the official website of Japan Meteorological Agency [3]. We select 11 places in a local region in the area of Gifu province. The geographical coordinates of these places are listed in Table 1. The temperature observation data is provided hour by hour. The number of timestamps is thus 24 in one day. The range of days is from the year 2005 to the year 2014 (ten years) and September 1 to September 30th (one month in each year). There are 7200 timestamps in total.

We set the location of the place $s0$ as (0,0) and create a Cartesian coordinate system, and assign the locations of the nearby places by converting the latitude and longitude into geometric distances in the coordinate system. The converted coordinates system is shown in Figure 2. The neighborhood relation is decided by a distance threshold (i.e., 30000) and is also shown in Figure 2. Table 2 lists the number of neighbor sensors of each sensor.

Table 1. Weather Dataset: Geographical Coordinates of Places

ID	City Name	(North latitude, East longitude)
$s0$	Kanayama	(35°39.7′,137°9.6′)
$s1$	Hagiwara	(35°53.5′,137°12.4′)
$s2$	Miyachi	(35°45.8′,137°17.3′)
$s3$	Kurogawa	(35°35.9′,137°19.1′)
$s4$	Nakatsugawa	(35°28.6′,137°29.2′)
$s5$	Ena	(35°26.8′,137°24.2′)
$s6$	Tajimi	(35°20.8′,137°6.5′)
$s7$	Minokamo	(35°26.7′,137°0.3′)
$s8$	Mino	(35°33.3′,136°54.6′)
$s9$	Yawata	(35°45.4′,136°58.7′)
$s10$	Nagataki	(35°55.4′,136°49.9′)

[3] http://www.data.jma.go.jp/gmd/risk/obsdl/index.php

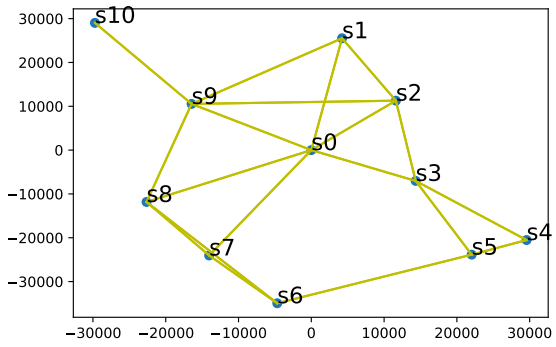


Fig. 2. Weather Dataset: Places and Neighborhood Relations

Table 2. Weather Dataset: Number of Neighbor Sensors

sensors	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
number of neighbors	6	3	4	4	2	3	3	3	4	5	1

4.2 Dataset: HVACS Evaluation House

This dataset is collected from the House-type HVACS (Heating, Ventilation and Air Conditioning System) Evaluation Facility built by Mitsubishi Electric R&D Centre Europe[4]. As one aspect of the smart house technology, we can install the sensors with various functions in a house. Figure 3 shows the outside and inside view of the smart house from which we collect the data. We install many temperature sensors in the whole house. In this paper, we use two groups of temperature sensors in the living room as two separated subsets, i.e. the sensors in the floor (HEH-Floor) and the sensors in the ceiling (HEH-Ceiling). There are eight sensors in each subset. The sensors record the temperature every 10 seconds. The data is collected in one day. There are 8619 records in the HEH-Floor dataset and 8618 records in the HEH-Ceiling dataset. Figure 4 shows the positions of sensors and the neighborhood relations among the sensors. The related positions of the sensors in the floor or ceiling are same.

Table 3. HVACS Evaluation House Dataset: Number of Neighbor Sensors

Dataset	sensors	s1	s2	s3	s4	s5	s6	s7	s8
HEH-Floor	number of neighbors	3	4	5	7	4	3	5	3
Dataset	sensors	s1	s2	s3	s4	s5	s6	s7	s8
HEH-Ceiling	number of neighbors	3	4	5	7	4	3	5	3

[4] http://www.mitsubishielectric.com/brief/randd/index.html

Fig. 3. Outside and Inside View of the HVACS Evaluation House

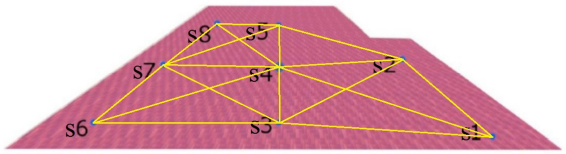


Fig. 4. HVACS Evaluation House Datasets (Floor or Ceiling): Positions and Neighborhood Relations

4.3 Experimental Settings

We utilize the following two evaluation metrics for evaluating the performance, i.e., Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). The formulas of these two metrics for a sensor s_k are as follows. We evaluate the average RMSE and MAE on all sensors. $\hat{y}_{ki}$ is the estimated value.

$$RMSE_k = \sqrt{\frac{1}{n_k}\sum_i (y_{ki} - \hat{y}_{ki})^2}, \qquad MAE_k = \frac{1}{n_k}\sum_i |y_{ki} - \hat{y}_{ki}|$$

We compare our *multitask* approach with the *independent* approach which is based on formulation (1) and only utilizes its own observation data as the features for learning, in the scenario of learning decentralized local models. In addition, we also show the performance of an ideal *centralized* approach as a reference. It is also based on formulation (1) but utilizes the observation data of all sensors as the features for learning.

We verify the performance of our approach on different window sizes. The window size t is set as {2,3,5,11,23} respectively for all datasets. Especially, for the weather dataset, $t = 23$ means using the observation data of previous hours in the length of one day to predict the temperature.

For our approach, the regularization hyperparameter λ is tuned by grid search with five-fold cross validation on the training set; the candidate values is in

Table 4. Experimental Results of Each Dataset; (10% Training, 90% Testing)

win.	RMSE				MAE			
size t	independ	multitask	improve	central	independ	multitask	improve	central
(a). Weather Dataset								
2	0.029693	0.029684	0.0312%	0.024120	0.021661	0.021631	0.1401%	0.017166
3	0.029296	0.029273	0.0779%	0.024309	0.021745	0.021709	0.1662%	0.017242
5	0.030002	0.029955	0.1578%	0.024574	0.022568	0.022508	0.2653%	0.017549
11	0.030982	0.030927	0.1748%	0.025636	0.023448	0.023384	0.2728%	0.018689
23	0.026703	0.026678	0.0924%	0.027869	0.019213	0.019179	0.1781%	0.020831
(b). HEH-Floor Dataset								
2	0.002112	0.002065	2.2689%	0.067664	0.001678	0.001640	2.2672%	0.053601
3	0.001965	0.001942	1.1907%	0.060207	0.001555	0.001537	1.1463%	0.047720
5	0.001904	0.001899	0.2744%	0.055196	0.001489	0.001486	0.2439%	0.043977
11	0.002088	0.002084	0.2027%	0.058846	0.001569	0.001567	0.1391%	0.047019
23	0.002821	0.002819	0.0603%	0.060079	0.002004	0.002003	0.0396%	0.047240
(c). HEH-Ceiling Dataset								
2	0.004019	0.003937	2.0369%	0.084777	0.003162	0.003095	2.1184%	0.065068
3	0.003765	0.003727	1.0020%	0.075965	0.002947	0.002916	1.0325%	0.058240
5	0.003615	0.003612	0.0718%	0.064683	0.002807	0.002805	0.0751%	0.049914
11	0.003770	0.003769	0.0110%	0.065543	0.002865	0.002865	-0.0009%	0.050197
23	0.004500	0.004490	0.2242%	0.064775	0.003280	0.003274	0.1743%	0.050013

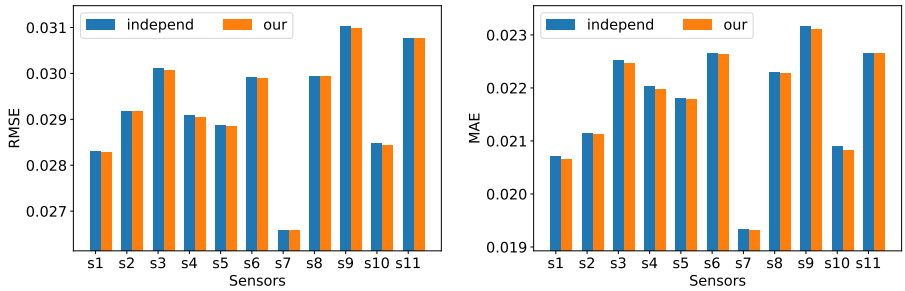
$\{0, 10^{-6}, 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$. The stop criterion θ is set to 10^{-6}. The maximum iteration number is set to 100.

The feature and target data of each instance is generated with overlap which can increase the number of instances and avoid the learned model overfitting to a chronological cycle. For example, assuming that window size $t = 2$, an instance is $(\mathbf{x}_{ki} = (x_{k(i-2)}, x_{k(i-1)}, 1),\ y_{ki} = x_{ki})$, then the next instance is $(\mathbf{x}_{k(i+1)} = (x_{k(i-1)}, x_{k(i)}, 1),\ y_{k(i+1)} = x_{k(i+1)})$.

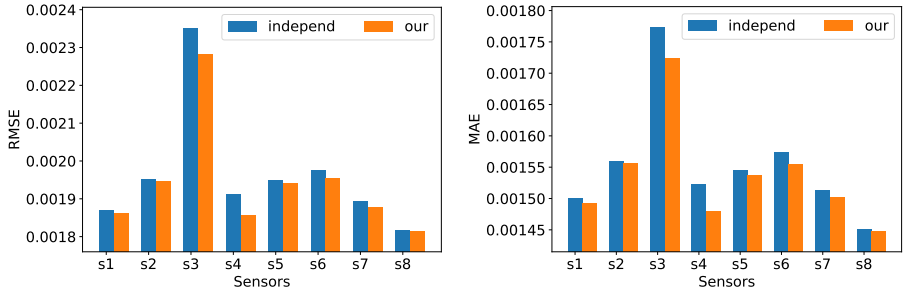
We split the training and testing sets with different rates of the entire data. In this paper, because we especially focus on the scenario that there is not lots of training data available, when using the testing data, we ignore the real timestamp of the instances. In other words, for an instance in the testing data, even if there are some observation data in the time interval between the training data and this instance, we do not use the data in this time interval to improve the model.

4.4 Experimental Results

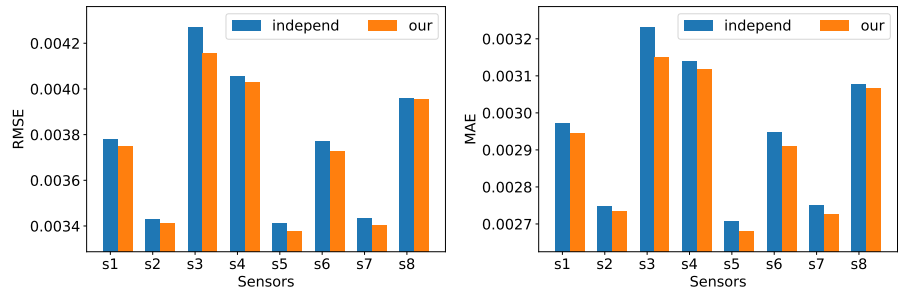
Table 4 lists the experimental results. To facilitate the presentation and highlight our contribution, we separately show the experimental results with the 10% split rate for the training set in the table. Considering the comparison between our multi-task approach and the independent approach, the *improve* column shows the improvement of our multi-task approach to the independent approach. It shows that out approach has better performance than the independent one on



(a) Weather Dataset



(b) HEH-Floor Dataset



(c) HEH-Ceiling Dataset

Fig. 5. Performance of Each Sensor. Window Size $t = 3$; (10% Training, 90% Testing)

all three datasets when the split rate for training set is low, which means that there is not lots of training data available.

Figure 5 shows the performance of each sensor respectively when the split rate for training is set to 10%. The window size is set to 3. It shows the robustness of our approach on each sensor, i.e., our approach not only improves the average performance of all sensors but also improves the performance of each sensor. Therefore, each sensor can obtain profits by using our approach.

Table 5 shows the experimental results on other split rate for training and testing sets. One of the observations is that in the cases of the high split rate for training set, e.g., 90% split rate for the training set, there are somewhat differences between the performance results of the weather dataset and the HVACS

Table 5. Experimental Results with Different Split Rates of Training Data

	win.	RMSE				MAE			
	size t	independ	multitask	improve	central	independ	multitask	improve	central
(a). Weather Dataset									
50% Train	2	0.029746	0.029744	0.0065%	0.023633	0.020934	0.020929	0.0230%	0.016556
	3	0.028881	0.028880	0.0048%	0.023585	0.020407	0.020403	0.0173%	0.016513
	5	0.028874	0.028872	0.0060%	0.023535	0.020453	0.020449	0.0198%	0.016482
	11	0.028719	0.028717	0.0066%	0.023420	0.020401	0.020397	0.0204%	0.016510
	23	0.026070	0.026069	0.0018%	0.022627	0.018233	0.018231	0.0135%	0.016044
90% Train	2	0.028348	0.028345	0.0130%	0.021877	0.020699	0.020695	0.0223%	0.016116
	3	0.027548	0.027545	0.0082%	0.021866	0.020094	0.020091	0.0153%	0.016142
	5	0.027578	0.027566	0.0414%	0.021823	0.020125	0.020109	0.0773%	0.016119
	11	0.027554	0.027542	0.0438%	0.021629	0.020198	0.020181	0.0860%	0.016091
	23	0.024576	0.024575	0.0035%	0.020461	0.017745	0.017743	0.0093%	0.015139
(b). HEH-Floor Dataset									
50% Train	2	0.001967	0.001966	0.0776%	0.005801	0.001568	0.001567	0.0777%	0.005297
	3	0.001865	0.001865	0.0278%	0.005098	0.001488	0.001488	0.0267%	0.004624
	5	0.001793	0.001793	0.0055%	0.004175	0.001431	0.001431	0.0053%	0.003734
	11	0.001795	0.001782	0.7256%	0.003554	0.001432	0.001422	0.7018%	0.003110
	23	0.001826	0.001779	2.5525%	0.002644	0.001461	0.001423	2.5784%	0.002217
90% Train	2	0.002011	0.002011	-0.0004%	0.002066	0.001608	0.001608	-0.0003%	0.001651
	3	0.001895	0.001895	-0.0089%	0.001932	0.001517	0.001518	-0.0074%	0.001547
	5	0.001822	0.001823	-0.0469%	0.001833	0.001457	0.001458	-0.0659%	0.001469
	11	0.001781	0.001786	-0.2562%	0.001746	0.001424	0.001428	-0.2930%	0.001397
	23	0.001764	0.001768	-0.2804%	0.001739	0.001410	0.001414	-0.2854%	0.001393
(c). HEH-Ceiling Dataset									
50% Train	2	0.003647	0.003647	0.0124%	0.006317	0.002905	0.002905	0.0145%	0.005473
	3	0.003459	0.003459	0.0000%	0.006060	0.002756	0.002756	0.0000%	0.005238
	5	0.003336	0.003336	0.0000%	0.005793	0.002655	0.002655	0.0000%	0.004986
	11	0.003298	0.003284	0.4034%	0.005135	0.002627	0.002617	0.3951%	0.004364
	23	0.003264	0.003243	0.6477%	0.004621	0.002602	0.002585	0.6407%	0.003859
90% Train	2	0.003494	0.003494	-0.0022%	0.003411	0.002776	0.002776	-0.0029%	0.002727
	3	0.003277	0.003277	0.0000%	0.003247	0.002603	0.002603	0.0000%	0.002595
	5	0.003160	0.003160	0.0000%	0.003130	0.002516	0.002516	0.0000%	0.002508
	11	0.003105	0.003126	-0.6876%	0.003034	0.002478	0.002494	-0.6426%	0.002430
	23	0.003087	0.003160	-2.3520%	0.002991	0.002460	0.002520	-2.4356%	0.002395

Evaluation House datasets. In the weather dataset, our approach still always outperforms the independent approach; while our approach is not better than the independent approach in the HVACS Evaluation House datasets. The reason is that the weather dataset is collected from a very large time period (ten years), while the HVACS Evaluation House datasets are collected from a short time period (one day). The time series data in the weather dataset is thus possible to be much more non-stationary. Even the split rate of the training set

is high, the independent regression model is still under-fitting to the weather data. In contrast, the independent regression model can fit the HVACS Evaluation House datasets well when there are many training data. In other words, our approach can outperform the independent approach when the bias of the independent model is high. Lack of enough training data is one of the reasons which can cause the under-fitting of the independent model.

The centralized approach is ideal and can use all observation data from all sensors for learning the models. Therefore, when there are plenty of training data, generally we expect that it can reach better performance than the independent approach and our distributed multi-task approach (e.g., weather dataset results in Table 4 and 5). The advantages and disadvantages of the centralized approach and the reasons for proposing decentralized approaches have been discussed at the beginning of the introduction section. In addition, because the candidate sensors are not selected and the model is not specialized for a given sensor, it is possible to have worse performance than our approach when the training data is not large enough for it. (e.g., HVACS Evaluation House datasets results in Table 4 and 5).

5 Conclusion

In this paper, we focus on the problem of learning decentralized local models for each sensor in a sensor network. We propose a novel distributed multi-task learning approach which incorporates neighborhood relations among sensors learn multiple models simultaneously in which each sensor corresponds to one task. Instead of broadcasting the observation data of a sensor in the entire network which is not satisfied for the reason of cost and privacy protection, this approach shares the model parameters as regularization terms in the objective function by assuming that neighbor sensors have similar model parameters. We conduct the experiments on three real datasets by predicting the temperature with the regression. They verify that our approach is effective, especially when the bias of an independent model which does not utilize the data from other sensors is high such as when there is not plenty of training data available.

In this paper, we select linear prediction because it is commonly used in practical cases. It would be useful to consider the solution for other types of complex models rather than linear prediction. On one hand, for many complex models which solve an optimization problem, we can add a regularization term of parameters in the object function like our approach. On the other hand, for complex models which cannot be solved in such manner, other specific topics need to be proposed, e.g., for clustering task and so on. We will address them in the future work. We assume that neighbor sensors have similar model parameters. If some neighbor sensors show a significant different behavior, e.g., the regularization term is high. It can be a useful clue for detecting fault sensors or unordinary places. These issues can be separated topics in the future work.

References

1. Ahmadi, M., Huang, Y., John, K.: Application of spatio-temporal clustering for predicting ground-level ozone pollution. In: Advances in Geocomputation, pp. 153–167. Springer (2017)
2. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. Advances in neural information processing systems 19, 41 (2007)
3. Beck, A., Tetruashvili, L.: On the convergence of block coordinate descent type methods. SIAM journal on Optimization 23(4), 2037–2060 (2013)
4. Caruana, R.: Multitask learning. Mach. Learn. 28(1), 41–75 (Jul 1997), http://dx.doi.org/10.1023/A:1007379606734
5. Ceci, M., Corizzo, R., Fumarola, F., Malerba, D., Rashkovska, A.: Predictive modeling of pv energy production: How to set up the learning task for a better prediction? IEEE Transactions on Industrial Informatics 13(3), 956–966 (2017)
6. Dinuzzo, F., Pillonetto, G., De Nicolao, G.: Client–server multitask learning from distributed datasets. IEEE Transactions on Neural Networks 22(2), 290–303 (2011)
7. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. Journal of Machine Learning Research 6(Apr), 615–637 (2005)
8. Evgeniou, T., Pontil, M.: Regularized multi–task learning. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 109–117. ACM (2004)
9. Gers, F.A., Eck, D., Schmidhuber, J.: Applying lstm to time series predictable through time-window approaches. In: International Conference on Artificial Neural Networks. pp. 669–676. Springer (2001)
10. Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., Madden, S.: Distributed regression: an efficient framework for modeling sensor network data. In: Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on. pp. 1–10. IEEE (2004)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
12. Low, Y., Gonzalez, J.E., Kyrola, A., Bickson, D., Guestrin, C.E., Hellerstein, J.: Graphlab: A new framework for parallel machine learning. arXiv preprint arXiv:1408.2041 (2014)
13. Sak, H., Senior, A.W., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Interspeech. pp. 338–342 (2014)
14. Shoham, Y., Powers, R., Grenager, T.: Multi-agent reinforcement learning: a critical survey. Tech. rep., Technical report, Stanford University (2003)
15. Tulone, D., Madden, S.: Paq: Time series forecasting for approximate query answering in sensor networks. In: European Workshop on Wireless Sensor Networks. pp. 21–37. Springer (2006)
16. Vanhaesebrouck, P., Bellet, A., Tommasi, M.: Decentralized collaborative learning of personalized models over networks. In: Artificial Intelligence and Statistics. pp. 509–517 (2017)
17. Wang, J., Kolar, M., Srebro, N., et al.: Distributed multi-task learning. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 751–760 (2016)