State-Driven Dynamic Sensor Selection and Prediction with State-Stacked Sparseness

Guo-Jun Qi[†], Charu Aggarwal[‡], Deepak Turaga[‡], Daby Sow[‡] and Phil Anno[§]

[†]Deptartment of Electrical Engineering and Computer Science,
University of Central Florida
4000 Central Florida Blvd, Orlando, FL 32816
guojun.qi@ucf.edu

[‡]IBM T.J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532
{charu,turaga,sowdaby}@us.ibm.com

[§]Geoscience and Reservoir Engineering Technology
ConocoPhillips, Houston, TX 77252
phil.d.anno@conocophillips.com

ABSTRACT

An important problem in large-scale sensor mining is that of selecting relevant sensors for prediction purposes. Selecting small subsets of sensors, also referred to as *active* sensors, often leads to lower operational costs, and it reduces the noise and information overload for prediction. Existing sensor selection and prediction models either select a set of sensors *a priori*, or they use adaptive algorithms to determine the most relevant sensors for prediction. Sensor data sets often show dynamically varying patterns, because of which it is suboptimal to select a fixed subset of active sensors. To address this problem, we develop a novel dynamic prediction model that uses the notion of *hidden system states* to dynamically select a varying subset of sensors. These hidden system states are automatically learned by our model in a data-driven manner. The proposed algorithm can rapidly switch between different sets of active sensors when the model detects the (periodic or intermittent) change in the system state. We derive the dynamic sensor selection strategy by minimizing the error rates in tracking and predicting sensor readings over time. We introduce the notion of *state-stacked sparseness* to select a subset of the most critical sensors as a function of evolving system state. We present experimental results on two real sensor datasets, corresponding to oil drilling rig sensors and intensive care unit (ICU) sensors, and demonstrate the superiority of our approach with respect to other models.

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

ACM 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783390.

1. INTRODUCTION

The deployment of sensor networks have become ubiquitous in many real-world systems to provide continuous monitoring [2][7][10]. Sensor networks are used to monitor dysfunctions in oil drilling rigs [9], and abnormal patient conditions in intensive care units (ICU) [17], and enemy activity in military scenarios. Indeed, our experimental results in this paper will explore some of these scenarios.

One of the pervasive properties of most sensor networks is that they are highly redundant in terms of the collected data. For example, a bird call at one sensor will typically also be detected by a sensor a few meters away. Even in cases where the sensors measure different quantities, they are strongly correlated by recording the same outside event. Therefore, a salient question arises as to whether one can retain only a smaller subset of sensors without losing a significant amount of information. An important practical motivation in such scenarios is to reduce the cost of operation and the information overload, while losing only a limited amount of information. For example, deploying, maintaining, and continuously using sensors in the earth subsurface's drilling environments is extremely expensive. By selecting a smaller set of sensors to track, significant direct and indirect cost savings are achieved. Moreover, depending on the nature of the correlations, all sensors are not equally useful to track. In many cases, including irrelevant sensors may even have negative qualitative effects in predictive applications.

Many state-of-the-art methods consider a static sensor selection model, in which a fixed set of sensors are actively selected. This paper deviates from these methods by developing a dynamic sensor selection model in which the set of selected sensors change over time as the hidden state of the system evolves. Such hidden states often have semantic interpretations in real scenarios. For example, in an oil drilling rig, the sensors have different patterns of correlations during periods when the drill is moving forward, or when the drill is moving backward. Furthermore, the presence

of "typical" dysfunctions or abnormal events will create different patterns of correlations. Similarly, in an ICU sensor network, the state of the patient might affect the underlying patterns of correlations between medical sensors. This implies that it is suboptimal to select a static correlation or selection model. Furthermore, in many application-specific scenarios, such as event detection, the system states might have important explanatory power, providing deep insight into the monitored events. Therefore, we view the approach in this paper as a first step to fully modeling the temporal sensor stream. This can provide a gateway to exploring other important problems in sensor network analysis.

In this paper, we develop a state-driven dynamic prediction model to track and predict the system states over time. These states are used to dynamically reveal the patterns of correlations between sensors, and select sets of non-redundant sensors as a function of changing states. Experimental results on the oil drilling rig and the ICU sensor networks demonstrate the competitive performance of the proposed algorithm.

The remainder of this paper is organized as follows. The problem is formulated in Section 3. The learning and inference algorithms are presented in Section 4. In 5, we present the details behind dynamic sensor selection and prediction algorithm, which can reduce to an ad-hoc static sensor selection model as a special case. The experiment results on the oil drilling sensor and the ICU sensor datasets are demonstrated in Section 6, followed by the conclusion in 7.

2. RELATED WORK

The analysis and modeling of the dynamics of sensor data streams has been extensively studied in the literature [16, 14, 1, 15, 6]. For this purpose, one of the most important problems is that of modeling the dynamic correlations between multiple sensor data streams over time in an effective and scalable fashion. The methods in [14, 16] model lag-correlations to forecast the future data streams over time. The techniques in [3, 8] develop a sensor selection model with a graph structure that models the domain-specific correlations between sensors. Aggarwal et al. [1] also develops a dynamic model for functional dependency between sensors in real time to predict the changing trends in the sensor data streams. The work in [20] uses linear regression to model the sensor correlations for prediction. An adaptive model [19] is proposed to predict the future sensor readings so that the unnecessary communications can be saved when the prediction model offers a satisfactory estimates of sensor measurements. The algorithm proposed in [4] develops adaptive querying based on pairwise covariances between sensor streams in order to reduce the communication overhead with limited bandwidth. In contrast, [5] proposes a hypothesis testing based prediction model that can bring the sensor sample series closer to weak stationarity to achieve low energy dissipation.

The aforementioned models and algorithms share a common characteristic – they reveal a dynamic model that is stationary in time. Although this is not a big obstacle to model the real world over a short time frame, the model will be unable to take advantage of the repetitions in the evolving trends over longer time frames. To address this deficiency, we propose a state-driven dynamic model for time-series sensor data. The states provide a model that can handle the typical forms of dynamic evolution over longer time frames. Any key sets of unique correlations over longer time periods can be essentially encoded and leveraged by these states.

3. FORMULATION

In this section, we will first introduce the formulation for the dynamic sensor selection problem.

3.1 Dynamic Sensor Selection

Consider a set of sensors indexed with $\mathcal{N} = \{1, 2, \cdots, N\}$. The nth sensor generates a real-valued time-series stream $x_t(n) \in \mathbb{R}$ at each time step t. The the sensor selection problem may be formulated as that of finding a small set of active sensors $\mathcal{M}_t \subseteq \mathcal{N}$ from which the readings of inactive sensors can be predicted as accurately as possible. Note that $\mathcal{M}_t$ is indexed with t to reflect the fact that active sensors are dynamically selected.

Definition 1 (Dynamic Sensor Selection). *Given a set of sensors $\mathcal{N} = \{1, 2, \cdots, N\}$, the dynamic sensor selection model is defined as that of selecting a smaller subset of sensors $\mathcal{M}_t \subseteq \mathcal{N}$ at each time step t, whose signals can accurately predict all sensor signals $\{x_{t'}(n)|n \in \mathcal{N}\}$ of both active and inactive sensors for $t' \geq t$.*

3.2 State Evolution Model

The aforementioned definition specifies a new paradigm of dynamic sensor selection model to track and predict sensor readings. The set of selected sensors can be dynamically adjusted as the system state evolves over time. For example, in the oil drilling application, different sets of active sensors should be selected to reflect the change in the mechanical and electrical state of the system as a consequence of changes in the nature of the of drilling operations. In many sensor applications, such states are not visible to the user, but they can only be perceived in terms of the changes in the patterns of underlying stream correlations. Therefore, we assume that these states are hidden, and they need to be learned and detected in a data-driven manner.

To model the impact of system conditions on sensor selection and prediction, we define the following hidden state evolution model for the system (e.g., an oil drilling rig or a patient) monitored by a sensor network $\mathcal{N}$.

Definition 2 (State Evolution Model). *A system has a set of K states denoted by $\mathcal{S} = \{1, 2, \cdots, K\}$. At each time t, the system has a hidden state $s_t \in \mathcal{S}$. The initial state of the system is modeled with a probability vector $\mathbf{\Pi} = [\pi_1, \pi_2, \cdots, \pi_K]^T$, i.e., $\Pr(s_0 = k|\mathbf{\Pi}) = \pi_k$. The system transits from state s_t to state s_{t+1} with a probability of $\mathbf{A}_{s_t, s_{t+1}}$, where $\mathbf{A}_{s_t, s_{t+1}}$ is an element of a transition matrix $\mathbf{A}$. Therefore, the overall state evolution model is fully defined by $\mathcal{E} = \{\mathcal{S}, \mathbf{\Pi}, \mathbf{A}\}$.*

This model defines a Markov chain of the states [11], where the state at each time only depends on the state at the preceding step. In other words, once the state is given for the current time, all the states of the future steps are independent of the past states. More complex higher-order Markov chains can be defined in which the system state depends on more previous states beyond the immediately previous one. However, increasing complexity can also cause overfitting. Therefore, for simplicity, we adopt the aforementioned state evolution model.

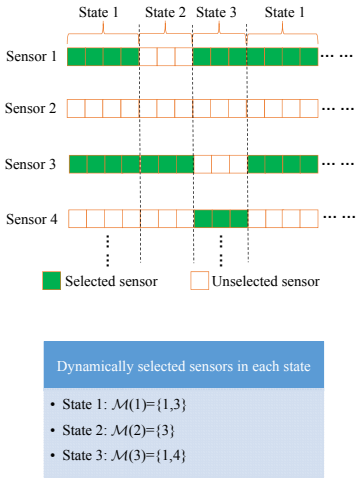


Figure 1: The dynamic sensor selection model.

3.3 State-Driven Sensor Selection and Prediction

Given the aforementioned state evolution model, the dynamic sensor selection model in Definition 1 can be re-casted as a problem of selecting a subset of active sensors $\mathcal{M}(s_t) \subseteq \mathcal{N}$ as a function of the system state s_t at each time t. In other words, different sets of active sensors are used to track and predict the streams of sensor data. This yields a novel dynamic sensor selection and prediction model as opposed to the static sensor selection model.

Figure 1 illustrates an example of dynamic sensor selection model. Four sensors are shown in the figure with 3 hidden states for the system. In each state, a distinct set of sensors are selected. For example, for state 1, the sensors in $\mathcal{M}(1) = \{1, 3\}$ are actively selected. Then, the system evolves into state 2, in which the sensor in $\mathcal{M}(2) = \{3\}$ is selected. The system keeps changing its state, and the corresponding set of sensors are selected along with each state.

We can define a P-step state-driven linear prediction model. It uses the signals collected from the active sensors to predict the signals of the (both active and inactive) sensors P steps into the future.

DEFINITION 3 (P-STEP MODEL WITH FIXED WINDOW R). *Given the states $\{s_{t-r}|0 \le r < R\}$ up to time t, the linear prediction model predicts $\mathbf{x}_{t+P} = [x_{t+P}(1), \cdots, x_{t+P}(N)]^T$ by*

$$\mathbf{x}_{t+P} = \sum_{r=0}^{R-1} \mathbf{W}(s_{t+P}, s_{t-r})\, \mathbf{x}_{t-r} + \mathbf{b}(s_{t+P}) + \varepsilon_{t+P} \qquad (1)$$

where $\mathbf{W}(s_{t+P}, s_{t-r}) \in \mathbb{R}^{N\times N}$ is a $N \times N$ prediction matrix from state s_{t-r} at time $t-r$ to state s_{t+P} at time $t+P$, and $\mathbf{b}(s_{t+P}) \in \mathbb{R}^N$ is the bias vector for the state s_{t+P} at time $t+P$. The notation ε_{t+P} represents the noise drawn from a Gaussian distribution $\mathcal{N}(\mathbf{0}_N, \sigma^2\mathbf{I}_N)$ with zero mean and isotropic variance σ^2. It is assumed that the noise terms at various time instants are independent.

Remark: In the aforementioned model, each prediction matrix $\mathbf{W}(s', s)$ makes use of sensor signals in state s to predict the future sensor readings in state s'. For this reason, we call the second argument s of this prediction matrix the *sender* state, and the first argument s_t as the *receiver* state. The use of both sender and receiver states to specify the prediction matrix enhances the model fidelity – the predicted sensor readings depend not only on the past state from which the sensor readings are collected, but also on the future state for which the prediction is made. This is in contrast to the Auto Regressive Model (ARM) [12], a popular dynamic model that predicts the sensor readings independent of the rapid-changing states.

Now consider a sender state s in (1), in which the columns of $\mathbf{W}(s', s)$ with all-zero entries correspond to the sensors that can be turned off without affecting the prediction result according to the model (1). Therefore, as long as all the matrices $\{\mathbf{W}(s', s)\,|s' \in \mathcal{S}\}$ with the same sender state s have a common set of zero columns, the corresponding sensors can be turned off when the sensor network runs into state s, no matter which receiver state it will run into in the future. Thus, those nonzero columns correspond to the selected sensors that ought to be turned on, which we denote by $\mathcal{M}(s)$.

OBSERVATION 1 (ZERO-COLUMN SENSOR SELECTION). *If all the prediction matrices $\mathcal{W}(s) \triangleq \{\mathbf{W}(s', s)|s' \in \mathcal{S}\}$ with the same sender state s have a common set of zero-columns, the corresponding sensors can be turned off without affecting the prediction result. In other words, we only need to select for each state s a set of sensors $\mathcal{M}(s)$ corresponding to those* common *nonzero columns for the prediction matrices in $\mathcal{W}(s)$.*

In the next section, we will see that how this requirement is fulfilled when we formulate sensor selection problem.

4. MODEL LEARNING AND INFERENCE

The general idea of solving a dynamic sensor selection and prediction problem is to implement an EM (Expectation-Maximization) algorithm, which alternates between inference of states $\{s_t|t = 0, 1, \cdots, T\}$ (E-Step) and the learning of the parameters including $\{\mathbf{W}(s', s)|(s', s) \in \mathcal{S}\times\mathcal{S}\}, \{\mathbf{b}(s)|s \in \mathcal{S}\}$ in the prediction model and $\{\Pi, A\}$ in the state evolution model (M-Step). In the following two subsections, we will show how to solve these two subproblems.

4.1 Model Learning: M-Step

Suppose we have a time series of sensor signals $\{\mathbf{x}_t|t = 0, 1, \cdots, T\}$, and we have sampled the corresponding sequence of states $\mathbf{s} = \{s_t|t = 0, 1, \cdots, T\}$. Now, we are ready to estimate the parameters for the state-evolution model in Definition 2 and the state-driven prediction model in Definition 3.

According to Eq. (1), with the independent Gaussian noise terms ε_t, the maximum likelihood criterion leads to the least squares problem for learning the prediction matrices $\{\mathbf{W}(s', s)|r, s', s \in \mathcal{S}\}$) by minimizing the following loss function:

$$\mathcal{L}_{ls} = \frac{1}{2\sigma^2} \sum_{t=P}^{T} \mathbb{E}_{\mathbf{s}} ||\mathbf{x}_{t+P} - \sum_{r=0}^{R-1} \mathbf{W}(s_{t+P}, s_{t-r})\, \mathbf{x}_{t-r} - \mathbf{b}(s_{t+P})||_2^2 \qquad (2)$$

Here, $\mathbb{E}_{\mathbf{s}}(\cdot)$ denotes the expectation over the hidden states $\mathbf{s} = \{s_t|t = 1, 2, \cdots\}$. This yields the smallest prediction errors on the training set in the least squares sense.

In addition, by the maximum likelihood criterion, the transition parameter $\mathbf{A}$ of the state evolution model can be solved as follows:

$$\mathbf{A}_{s,s'} = \frac{\sum_{t=0}^{T-1} \mathbb{E}_{s_t,s_{t+1}} \delta\left[\!\left[s_t = s, s_{t+1} = s'\right]\!\right]}{T} \tag{3}$$

Here, $\delta\left[\!\left[\cdot\right]\!\right]$ is the indicator function, which outputs 1 if the included condition is satisfied and 0, otherwise. This equation counts how many times a transition from s to s' has occurred in expectation sense.

In Section 4.2, we will present how to sample a sequence of states $\{s_t | t = 1, 2, \cdots, T\}$, which are used to approximate the aforementioned expectations.

4.1.1 Dynamic Sensor Selection

To select a smaller set of sensors for each state, we introduce the following $L_{1,2}$ norm which encourages zero-columns on the prediction matrix.

DEFINITION 4 ($L_{1,2}$ MATRIX NORM). *The $L_{1,2}$ norm for a matrix M is defined as the sum of L_2 norms of the column vectors for this matrix, i.e.,*

$$\|\mathbf{M}\|_{1,2} = \sum_{i=1}^{N} \|\mathbf{M}_i\|_2 \tag{4}$$

Here, $\mathbf{M}_i$ is the ith column vector of $\mathbf{M}$.

The minimization of an objective function with such $L_{1,2}$ norms on a matrix tends to make some of entire columns have zero elements. Forcing such sparseness with carefully tailored regularization is achieved in many data mining and machine learning applications such as *Lasso* [21]. Just as *Lasso* sparsity is used for feature selection, we use this form of regularization with the following *stacked* sparseness for sensor selection.

The goal is to fulfill the zero-column sensor selection requirement established in Observation 1. We can stack all the matrices corresponding to the same sender state s, which forms a bigger matrix as follows:

$$\boldsymbol{\mathcal{W}}(s) = \begin{bmatrix} \mathbf{W}(1,s) \\ \cdots \\ \mathbf{W}(k,s) \\ \cdots \\ \mathbf{W}(K,s) \end{bmatrix} \in \mathbb{R}^{KN \times N} \tag{5}$$

Since each column of $\boldsymbol{\mathcal{W}}(s)$ is a concatenation of the columns of all the matrices with the same sender state s, minimizing $\|\boldsymbol{\mathcal{W}}(s)\|_{1,2}$ can force some columns of these matrices to be zero. This results in common zero columns, which is referred to as *stacked sparseness* for the common sender state in this paper. Then, we only need to select the sensors corresponding to nonzero columns at state s and leverage their measurements to predict the future sensor readings, without affecting the prediction model.

Putting together the least square loss and $\|\boldsymbol{\mathcal{W}}(s)\|_{1,2}$, we have the following minimization problem [1]:

$$\min_{W} \mathcal{L}(\boldsymbol{\mathcal{W}}) \triangleq \mathcal{L}_{ls}(\boldsymbol{\mathcal{W}}) + \mu \sum_{s=1}^{K} \|\boldsymbol{\mathcal{W}}(s)\|_{1,2} \tag{6}$$

[1]Note that the objective is a function of both the prediction matrices $\boldsymbol{\mathcal{W}}$ and the bias vectors $\{\mathbf{b}(s) | s \in \mathcal{S}\}$. Since it is much easier to optimize with respect to the bias vectors, we concentrate on $\boldsymbol{\mathcal{W}}$ and discard the bias vectors in the argument. The discussion about optimizing with respect to $\{\mathbf{b}(s) | s \in \mathcal{S}\}$ can be found after Theorem 1.

Here, $\boldsymbol{\mathcal{W}}$ is a matrix that concatenates all the prediction matrices following the notation in (5), i.e.,

$$\boldsymbol{\mathcal{W}} \triangleq [\boldsymbol{\mathcal{W}}(1), \boldsymbol{\mathcal{W}}(2), \cdots, \boldsymbol{\mathcal{W}}(K)] \in \mathbb{R}^{KN \times KN},$$

and μ is a positive balancing parameter that trades off between the loss function and the $L_{1,2}$ norm. A large value of μ will result in a smaller set of selected sensors for each state. This parameter is set by balancing between the required level of training accuracy and the number of sensors that should be maintained in the field.

4.1.2 Cost-Sensitive Sensor Selection

We can also consider the scenario where the costs of maintaining different types of sensors are unequal in the field. In this case, associated with each sensor $n \in \mathcal{N}$, we can explicitly assign a positive weight δ_n to represent the cost. This weight can be multiplied with each column of $\boldsymbol{\mathcal{W}}(s)$. Then, minimizing the $L_{1,2}$ norm of this cost-weighted matrix results in de-selection of the sensors with larger cost weights by zeroing out the corresponding columns rather than the sensors with smaller cost weights. The $L_{1,2}$ norm of the cost-weighted matrix can be written in a compact matrix form as

$$\|\boldsymbol{\mathcal{W}}(s) \cdot \mathbf{diag}(\delta_1, \cdots, \delta_N)\|_{1,2}$$

with $\mathbf{diag}(\cdot)$ denoting a diagonal matrix with the cost weights as its elements.

4.1.3 Optimization Algorithm

This section provides the details of finding the optimal solution to the objective function (6). It is nontrivial to optimize this objective function over $\boldsymbol{\mathcal{W}}$ with a non-smooth $L_{1,2}$ norm term, since the conventional gradient descent method is ineffective in such cases. For example, the sub-gradient decent can converge very slowly and the generated minimizer may not fully exploit the sparseness property associated with the $L_{1,2}$ norm. Thus, we present an alternative gradient-based algorithm which can handle such cases. If desired, the reader can skip the following derivation, and go to Algorithm 1 directly.

At a broad level, we adopt the proximal gradient family of algorithms. It is an optimization method that solves the original problem via a sequence of approximations. At the current estimate of $\boldsymbol{\mathcal{W}}^{(l)}$ at each iteration l, we expand the first-order Taylor series of $\mathcal{L}_{ls}(\boldsymbol{\mathcal{W}})$. Then, the objective function (2) becomes the following:

$$\begin{aligned} \mathcal{Q}_\tau(\boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{W}}^{(l)}) &\triangleq \mathcal{L}_{ls}(\boldsymbol{\mathcal{W}}^{(l)}) + \langle \nabla \mathcal{L}_{ls}(\boldsymbol{\mathcal{W}}^{(l)}), \boldsymbol{\mathcal{W}} - \boldsymbol{\mathcal{W}}^{(l)} \rangle \\ &\quad + \frac{\tau}{2} \|\boldsymbol{\mathcal{W}} - \boldsymbol{\mathcal{W}}^{(l)}\|_F^2 + \mu \sum_{k=1}^{K} \|\boldsymbol{\mathcal{W}}(s)\|_{1,2} \\ &= \frac{\tau}{2} \|\boldsymbol{\mathcal{W}} - \boldsymbol{\mathcal{G}}_\tau^{(l)}\|_F^2 + \mu \sum_{k=1}^{K} \|\boldsymbol{\mathcal{W}}(s)\|_{1,2} + \text{const} \end{aligned} \tag{7}$$

Here, $\nabla \mathcal{L}_{ls}(\boldsymbol{\mathcal{W}}^{(l)})$ is the gradient of the loss function $\mathcal{L}_{ls}(\boldsymbol{\mathcal{W}})$ at $\boldsymbol{\mathcal{W}}^{(l)}$, $\|\cdot\|_F$ is the Frobenius norm, and const is a term irrespective of $\boldsymbol{\mathcal{W}}$:

$$\boldsymbol{\mathcal{G}}_\tau^{(l)} = \boldsymbol{\mathcal{W}}^{(l)} - \tau^{-1} \nabla \mathcal{L}_{ls}(\boldsymbol{\mathcal{W}}^{(l)}) \tag{8}$$

Here, τ is a parameter that weights the quadratic term in (7), which is usually set to a larger value than the Lipschitz constant of the least squares loss $\mathcal{L}_{ls}(\boldsymbol{\mathcal{W}})$.

Algorithm 1 Optimization Algorithm for Problem (6)

input Choose $\boldsymbol{\mathcal{W}}^{(0)} = \boldsymbol{\mathcal{W}}^{(-1)}$, $t^{(0)} = t^{(-1)} = 1$.
  for $l = 0, 1, 2, \cdots$ **do**
    // generate $\boldsymbol{\mathcal{W}}^{(l+1)}$ from $\boldsymbol{\mathcal{W}}^{(l)}$ according to the following iterations.
    Step 1 Set $\boldsymbol{\mathcal{Y}}^{(l)} = \boldsymbol{\mathcal{W}}^{(l)} + \frac{t^{(l-1)} - 1}{t^{(l)}}(\boldsymbol{\mathcal{W}}^{(l)} - \boldsymbol{\mathcal{W}}^{(l-1)})$;
    Step 2 Set $\boldsymbol{\mathcal{G}}_{\tau}^{(l)} = \boldsymbol{\mathcal{Y}}^{(l)} - \tau^{-1}\nabla\mathcal{L}_{ls}(\boldsymbol{\mathcal{Y}}^{(l)})$;
    Step 3 Solve $\boldsymbol{\mathcal{W}}^{(l+1)}$ as in Eq. (10);
    // update the bias vector $\mathbf{b}^{(l)}$
    Step 4 Set $\mathbf{d}^{(l)} = \mathbf{b}^{(l)} + \frac{t^{(l-1)} - 1}{t^{(l)}}(\mathbf{b}^{(l)} - \mathbf{b}^{(l-1)})$;
    Step 5 Set $\mathbf{b}^{(l+1)} = \mathbf{d}^{(l)} - \tau^{-1}\frac{\partial\mathcal{L}_{ls}(\mathbf{d}^{(l)})}{\partial\mathbf{b}}$;
    // update the interpolation parameter
    Step 6 Set $t^{(l+1)} = \frac{1 + \sqrt{1 + 4(t^{(l)})^2}}{2}$.
  end for

Since the least squares loss $\mathcal{L}_{ls}$ is a differentiable function, its gradient $\nabla\mathcal{L}_{ls}(\boldsymbol{\mathcal{W}}^{(l)})$ in Eq. (8) can be derived easily.

Note that $\mathcal{Q}_{\tau}(\boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{W}}^{(l)})$ is a convex function of $\boldsymbol{\mathcal{W}}$ and thus it has a minimizer, yielding $\boldsymbol{\mathcal{W}}^{(l+1)}$ for the next iteration as follows:

$$\boldsymbol{\mathcal{W}}^{(l+1)} = \arg\min_{\boldsymbol{\mathcal{W}}} \mathcal{Q}_{\tau}(\boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{W}}^{(l)}) \tag{9}$$

Fortunately, Problem (9) has a closed-form solution. Denote the ith column of $\boldsymbol{\mathcal{W}}^{(l+1)}$ by $[\boldsymbol{\mathcal{W}}^{(l+1)}]_{:,i}$. Then, we have:

$$\left[\boldsymbol{\mathcal{W}}^{(l+1)}\right]_{:,i} = \left(1 - \frac{\mu}{\tau\left\|\left[\boldsymbol{\mathcal{G}}_{\tau}^{(l)}\right]\right\|_{:,i}}\right)_{+}\left[\boldsymbol{\mathcal{G}}_{\tau}^{(l)}\right]_{:,i} \tag{10}$$

Here, $[\boldsymbol{\mathcal{G}}_{\tau}^{(l)}]_{:,i}$ is the ith column of $\boldsymbol{\mathcal{G}}_{\tau}^{(l)}$ defined in Eq. (8), and $(z)_{+}$ outputs the positive component of z, i.e., it outputs z if $z > 0$, and 0. otherwise. At each iteration, this minimizer tends to yield zero-columns corresponding to the sensors that can be excluded for each state.

When the cost weights $\delta_n, n \in \mathcal{N}$ are considered, the aforementioned update should be changed to

$$\left[\boldsymbol{\mathcal{W}}^{(l+1)}\right]_{:,i} = \left(1 - \frac{\delta_n\mu}{\tau\left\|\left[\boldsymbol{\mathcal{G}}_{\tau}^{(l)}\right]\right\|_{:,i}}\right)_{+}\left[\boldsymbol{\mathcal{G}}_{\tau}^{(l)}\right]_{:,i} \tag{11}$$

Therefore, the column for a sensor with larger δ_n is more likely to be zeroed out.

Algorithm 1 summarizes the optimization approach for Problem (2). It is noteworthy that in Step 1, we compute $\boldsymbol{\mathcal{Y}}^{(l)}$ that plays the same role of $\boldsymbol{\mathcal{W}}^{(l)}$ in Step 2-3 as in the aforementioned derivation. Here, $\boldsymbol{\mathcal{Y}}^{(l)}$ interpolates between the solutions in two successive steps. Such a Nesterov optimization strategy [13] can accelerate the convergence of Algorithm 1 to a rate of $O(1/L^2)$ where L is the number of the iterations. The following theorem formally states this result.

THEOREM 1 (CONVERGENCE). *Let $\{\boldsymbol{\mathcal{W}}^{l}\}$ be the sequence generated by Algorithm 1. Then for any $\boldsymbol{\mathcal{W}}^{\star}$ with $\mathcal{L}(\boldsymbol{\mathcal{W}}^{\star}) \leq \inf_{\boldsymbol{\mathcal{W}}} \mathcal{L}(\boldsymbol{\mathcal{W}}) + \epsilon$, we have*

$$\min_{l=0,1,\cdots,L+1} \mathcal{L}(\boldsymbol{\mathcal{W}}^{(l)}) \leq \mathcal{L}(\boldsymbol{\mathcal{W}}^{\star}) + \frac{4C_f\|\boldsymbol{\mathcal{W}}^{\star} - \boldsymbol{\mathcal{W}}^{(0)}\|_F^2}{(L+2)^2}$$

where C_f is the Lipschitz constant of $\mathcal{L}_{ls}(\boldsymbol{\mathcal{W}})$.

Remark: The above theorem can be proved in straightforward way with a similar idea to that in [18]. The detail is omitted here due to space limitations. This theorem shows the convergence of Algorithm 1 to an ϵ-optimal minimizer $\boldsymbol{\mathcal{W}}^{\star}$.

It is worth noting that the optimization with respect to the bias vectors $\mathbf{b} = [\mathbf{b}(1), \mathbf{b}(2), \cdots, \mathbf{b}(K)]$ can be easily obtained since the objective function is differentiable with respect to them. Note that the non-smooth $L_{1,2}$ norm term is independent of the bias vectors. Thus, the conventional gradient decent algorithm is directly applicable as in Steps 4-5 in Algorithm 1 with a similar interpolation strategy.

4.2 Sampling State Sequence

In this section, we present a sequential sampling algorithm to sample the sequence of states $\{s_t | t = 0, 1, \cdots, T\}$ which are used to approximate the expectation in Eq. (2).

Given the current estimate of model parameters, the past states $s_{1:t-1} = \{s_1, s_2, \cdots, s_{t-1}\}$ and the observations $\mathbf{x}_{1:t} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\}$ up to time t, the posterior probability for state s_t at step t can be written as

$$\Pr(s_t | s_{1:t-1}, \mathbf{x}_{1:t}) \propto \mathbf{A}_{s_{t-1}, s_t} \cdot \exp\left(-\frac{1}{2\sigma^2}||\mathbf{x}_t - \sum_{r=0}^{R-1} \mathbf{W}(s_t, s_{t-P+r})\,\mathbf{x}_{t-P+r} - \mathbf{b}(s_t)||_2^2\right) \tag{12}$$

The first factor on the right-hand side of the equation is the prior transition probability between two consecutive states, and the second factor is the likelihood proportional to the Gaussian distribution over the remainder of the least square prediction error. Accordingly, s_t can be sampled according to this posterior distribution directly. This process is iterated over t from 1 to T. Algorithm 2 summarizes the algorithm of sampling the sequence of states.

Algorithm 2 Sampling State Sequence

input a sequence of sensor measurements $\{\mathbf{x}_t | t = 1, 2, \cdots, T\}$
  Sample s_0 according to Π;
  for $t = 1, 2, \cdots, T$ **do**
    Sample s_t according to Eq. (12).
  end for

It is worth noting that we can handle missing sensor measurements when sampling the state at a time step t, by only considering the observed entries in $\mathbf{x}_t$ to compute the $\|\cdot\|_2$ term in the exponent since the unobserved entries can be marginalized out for a multivariate Gaussian distribution defined by Eq. (1). This is important because sensor observations are traditionally noisy with many missing values.

Algorithm 3 Learning Model Parameter

input a sequence of sensor measurements $\{\mathbf{x}_t|t = 1, 2, \cdots, T\}$
Initialize the model parameter
// Iterate between M-Step, and E-Step.
E-Step Sample the sequence of the states according to Algorithm 2.
M-Step Apply Algorithm 1 to update the current estimate of model parameter;

Algorithm 4 Dynamic Sensor Selection and Prediction

input the learned model parameter.
for each step t **do**
Sample the current state s_t according to Eq. (12);
if the new state differs from s_{t-1} **then**
Sensor selection: Turn on the sensors in $\mathcal{M}(s_t)$ corresponding to the nonzero columns in $\boldsymbol{\mathcal{W}}(s_t)$; the other sensors not in $\mathcal{M}(s_t)$ can be turned off.
end if
Data prediction: Apply the model in (1) to predict the sensor measurements x_{t+P} for P steps ahead of time.
end for

Additionally, in the learning phase, we also need to sample the posterior distribution over $s_{t'}$ with $t' < t$

$$\Pr(s_{t'}|s_{1:t}, \mathbf{x}_{1:t}) \propto \mathbf{A}_{s_{t'-1}, s_{t'}} \mathbf{A}_{s_{t'}, s_{t'+1}} \cdot \prod_{p=0}^{P} \exp(-\frac{1}{2\sigma^2} ||\mathbf{x}_{t'+p} - \sum_{r=0}^{R-1} \mathbf{W}\left(s_{t'+p}, s_{t'+p-P-r}\right) \mathbf{x}_{t'+p-P-r} - \mathbf{b}(s_{t'+p})||_2^2) \tag{13}$$

In this case, we need to consider both the information of the past $t < t'$ and the future $t > t'$ time steps over a sequence of states. Specifically, the first two terms on the right-hand side are the transition probabilities to and from $s_{t'}$ of the current state, and each term in the product is the likelihood proportional to the Gaussian distribution on the least-square remainder from time t' to time $t' + P$.

The sampling process can be performed iteratively to sample the states a-posteriori given the sensor measurements up to time t, and the sampled sequence $s_{1:t}$ can be used to approximate the expectation in Eq. (2)-(3). Markov Chain Monte Carlo (MCMC) method provides theoretical guarantee on this sampling-based approximate.

4.3 Summary

Algorithm 3 summarizes the algorithm of learning the model parameters. Given the sampled states $\{s_t|t = 0, 1, \cdots, T\}$, we apply Algorithm 1 to update the current estimate of model parameters. Given the current model parameters, we sample the sequence of the states according to Algorithm 2. We iterate between these two subroutines of algorithms to find the optimal model parameters.

5. DYNAMIC MODEL

With the learned model from the previous section, we can apply it to dynamically select the sensors and predict the sensor data streams ahead of time. We summarize our approach to dynamic sensor selection and prediction in Algorithm 4.

For dynamic sensor selection, at each time t, if the current state changes from the previous one, the set of active sensors should also change accordingly. Those sensors that are not in the active set $\mathcal{M}(s_t)$ can be turned off without affecting the prediction of the future sensor measurements. This is also discussed in the previous section.

At each step t, the model is applied to predict the sensor measurements $\widehat{x}_{t+P}$ at time $t + P$, i.e., P-step ahead of the current time t, by substituting $t+P$ for t in Eq. (1). Specifically, we take the expectation of x_{t+P} over the state s_{t+P} at time $t + P$,

$$\begin{aligned}\widehat{x}_{t+P} &\triangleq \mathop{\mathbb{E}}_{s_{t+P}} [\mathbf{x}_{t+P}|s_{1:t}] \\ &= \sum_{r=0}^{R-1} \mathop{\mathbb{E}}_{s_{t+P}} [\mathbf{W}\left(s_{t+P}, s_{t-r}\right) \mathbf{x}_{t-r}|s_{1:t}] + \mathop{\mathbb{E}}_{s_{t+P}} [\mathbf{b}(s_{t+P})|s_{1:t}]\end{aligned} \tag{14}$$

Here, given the states $s_{1:t}$ up to time t, the distribution of state s_{t+P} only depends on s_t according to the Markovian property of the state-evolution model, i.e.

$$\Pr(s_{t+P}|s_{1:t}) = \Pr(s_{t+P}|s_t) = [\mathbf{A}^P]_{s_t, s_{t+P}} \tag{15}$$

where $\mathbf{A}$ is the transition matrix, and $[\mathbf{A}^P]_{s_t, s_{t+P}}$ is the entry of $\mathbf{A}^P$ corresponding to the sender state s_{t-r} and the receiver state s_{t+P}. Then, we have:

$$\begin{aligned}&\mathop{\mathbb{E}}_{s_{t+P}} [\mathbf{W}\left(s_{t+P}, s_{t-r}\right) \mathbf{x}_{t-r}|s_{1:t}] \\ &= \sum_{s_{t+P} \in \mathcal{S}} \Pr(s_{t+P}|s_{1:t}) \mathbf{W}\left(s_{t+P}, s_{t-r}\right) \mathbf{x}_{t-r}\end{aligned} \tag{16}$$

and

$$\mathop{\mathbb{E}}_{s_{t+P}} [\mathbf{b}(s_{t+P})|s_{1:t}] = \sum_{s_{t+P} \in \mathcal{S}} \Pr(s_{t+P}|s_{1:t}) \mathbf{b}(s_{t+P}) \tag{17}$$

Plugging Eq. (16)-(17) back into Eq. (14), we get the estimated sensor measurement at time $t + P$.

5.1 A Special Case : Static Sensor Selection

So far, we have focused on a dynamic sensor selection model, in which different sets of sensors can be selected for different states. However, in some applications, we may be interested in a static sensor selection model, where a fixed set of sensors are selected no matter in which state the sensor network might be in. For this purpose, we can stack all the prediction matrices in the column direction, and apply a $L_{1,2}$ norm to enforce all the matrices have the same set of zero-columns. Then, the sensors corresponding to these zero-columns can be excluded without affecting the prediction of the future sensor readings. In this spirit, the static sensor selection is a special case of its dynamic counterpart. The corresponding optimization algorithm does not need to change much to handle the similar stacked $L_{1,2}$ matrix norm.

Note that, although this is a static sensor selection model, we can still use the state evolution model in Definition 1 to capture the dynamics of the sensor network, and the state-driven prediction model (1) to dynamically predict the sensor signals. The only difference is that the set of selected sensors will not change as the states evolve. In the experiment, we will compare between the dynamic and the static model.

Table 1: ICU sensors for monitoring patient health conditions.

sensor id	abbreviation	measurement
1	Hr	heart rate
2	Temp	body temperature
3	SpO2	saturation of peripheral oxygen
4	BPd	diastolic blood pressure
5	BPs	systolic blood pressure
6	BPm	mean blood pressure
7	RESP	respiration

6. EXPERIMENTS

In this section, we conduct experiments to test the proposed sensor selection and prediction model with two real sensor networks. First, we will introduce the data sets collected from these two sensor networks. We will explain how they are collected and some statistics about them.

6.1 Data Sets

Figure 2 shows three examples of a pair of sensor data time-series (top row) and their correlations (bottom row) extracted from the two datasets. The datasets exhibit both periodic and gradually-changing patterns of the correlations between sensors.

6.1.1 *Oil drilling rig sensor network*

This sensor network is deployed on oil rigs, and is used is used to monitor the status of the oil drilling system, and diagnosing and predicting the potential risk of dysfunction. The sensor network consists of sensors deployed at the surface, as well as sensors along the drill string, that operate within the wellbore in the subsurface of the earth. As the drill moves for vertical and sometimes horizontal drilling, and encounters different formations, the position and properties of these sensors with relation to the earth's surface changes - affecting the observed measurements, and their correlations.

We consider a set of 33 such surface and wellbore sensors and demonstrate how our state based sensor selection approach can enable significant reductions in the number of sensors needed, while minimizing prediction error.

6.1.2 *ICU sensor network*

The second data set is generated by medical sensors used to monitor the status of patients in ICUs. We collected the readings of seven different types of ICU sensors (see Table 1) from 357 patients, which closely monitor the health conditions ranging from heart rate, blood pressure to respiration. For different patients, the length of sensor measurements vary considerably. Some patients only have a few hundred measurements for each sensor, while others have several million measurements. We wished to track and predict the main trends of each patient's states, and select a set of sensors whose readings are the most critical to predict these states. It is often beneficial to understand which sensors contain key indications on the health condition of the monitored patient in ICUs.

6.2 Results on Oil Drilling Rig Sensors

Figure 3 presents the comparison of error rates among the following algorithms:

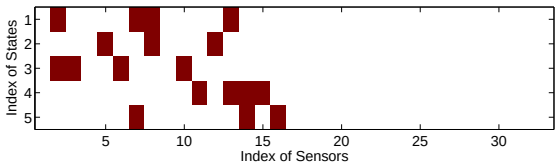


Figure 4: Active sensors in each state for oil drilling rig sensor dataset: the red cells in each row represent the active sensors for a state.

- Sampled Uni-ARM: the baseline benchmark. This model uses an auto-regressive model (ARM) [12] to predict the future sensor readings, and a univariate ARM is created to model each individual sensor. Also, at each time step, a subset of sensors is selected in random.
- Sampled Multi-ARM: this model is the same as Sampled Uni-ARM, except that a multi-variate ARM is created to model the sensors simultaneously.
- Uni-ARM PES: this model uses a univariate model for each individual sensor, but a power-efficient selection (PES) criterion is adopted to select a subset of the most critical sensors that can provide the best accurate prediction over time on the future readings [1].
- Multi-ARM PES: It is the same as Uni-ARM PES, except a multi-variate ARM is used to model the dynamics of sensor readings over time [1].
- SDSSP (State-driven sensor selection and prediction): this is the proposed method in this paper.

In Figure 3(a), the error rates are compared versus the varying the number of active sensors selected by different algorithms. Since the proposed SDSSP is a dynamic sensor selection model, the number of active sensors keep changing over time with the evolving states, so that the number of active sensors in the figure is an average number of sensors selected by the states over time. The other algorithms are compared by setting the numbers of active sensors to integers nearest to these average numbers. Figures 3(b)-3(c) compare the performances with varying window sizes R and prediction lags P as in Eq. (1). The results show the proposed SDSSP consistently outperforms the other methods.

Figure 4 shows the active sensors in each state derived from the oil drilling sensor dataset. The sensors indexed from $1-8$ and $20-33$ are global sensors that are deployed on the surface of the rig platform to monitor the operation of the whole oil drilling system. On the other hand, the sensors indexed from $9-19$ are deployed in the vertical or curved-lateral segment of the wellbore. We obtain the active sensors by setting the cost for each undersurface sensor to twice that for each global sensor. All five states, except state 4, select both global and vertical sensors; on the contrary, state 4 only selects the vertical sensors, implying that this state corresponds to a drilling operation in the vertical borehole reaching deep into the earth's subsurface.

Figure 5 compares the error rates between the dynamic model (SDSSP) and the static model as presented in Section 5.1 where a fixed set of sensors are selected. The comparison is made by varying the window size R and the prediction lag P. Clearly, the dynamic model usually provides more

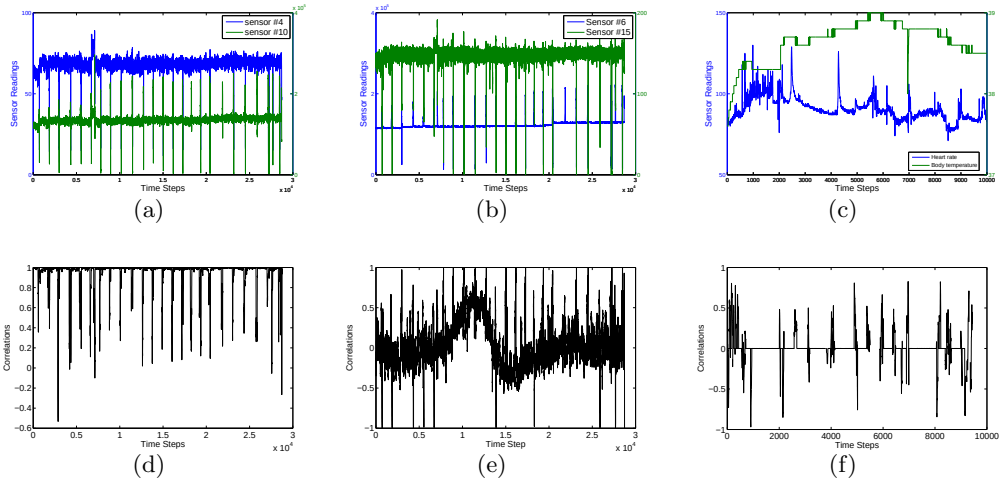
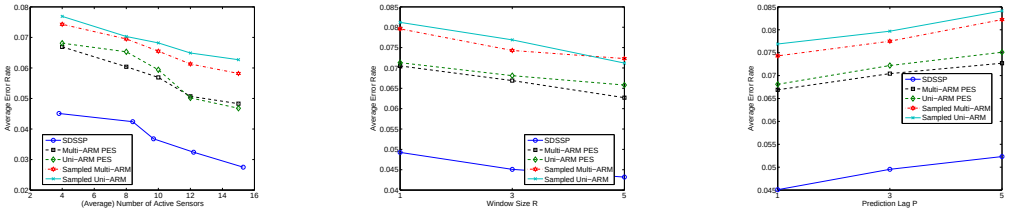


Figure 2: Examples of oil rig sensor dataset (first two columns) and ICU sensor dataset (the last column). Subfigure (a)-(c): a pair of sequences of sensor readings; Subfigure (d)-(f): the sequence of Pearson's correlation between the two sensors.



(a) Prediction Lag $P = 1$, Window Size $R = 3$ (b) Prediction Lag $P = 1$ (c) Window Size $R = 3$

Figure 3: Comparison of error rates by the different algorithms on the oil drilling rig sensor dataset: (a) the error rates with varying numbers of active sensors (achieved with $P = 1$ and $R = 3$); (b) the error rates with varying window sizes R when the prediction lag is fixed to $P = 1$; (c) the error rates with varying prediction lags P when window size is fixed to $R = 3$.

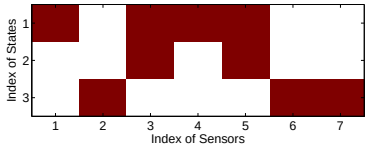


Figure 7: Active sensors in each state for ICU sensor dataset: the red cells in each row represent the active sensors for a state.

accurate prediction on the sensor data streams over time than its static counterpart.

6.3 Results on ICU Sensors

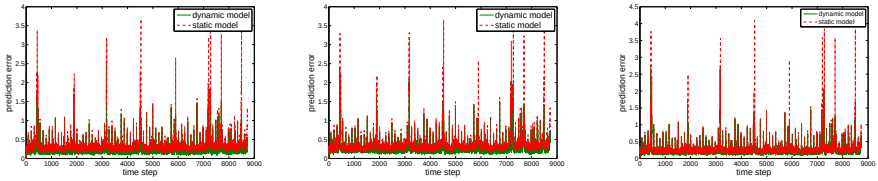
Similar to the experimental set up for oil drilling rig sensors, Figure 6 presents the results in comparison with other algorithms. Through the comparison, SDSSP outperforms the other algorithm on this ICU sensor dataset in terms of different number of active sensors, varying window sizes and prediction lags.

Figure 7 illustrates the active ICU sensors in each state, where the indices of these ICU sensors have been shown in Table 1. We can find that for the first two states, the selected sensors usually measure the instant conditions of the patients, such as heart rates and diastolic/systolic blood pressure. On the other hand, the third state selects the sensors that are relevant to a relatively long-time health conditions, including mean blood pressure and body temperature. In accordance with these findings, we observe that when the monitored patient is in a stable condition (i.e., the sensor readings do not change much), state 3 is often activated; otherwise, patients are more likely in instant state 1 or state 2. This is yet another example of the kind of semantic interpretation that one can often associate with the hidden states that are discovered in a data-driven manner.

Figure 8 compares the proposed dynamic model with the static counterpart on seven different ICU sensors with the window size P and prediction lag R both set to 1. Most of time, the dynamic model achieves better performance than the static model on these seven ICU sensors.

6.4 Effect of the Number of States

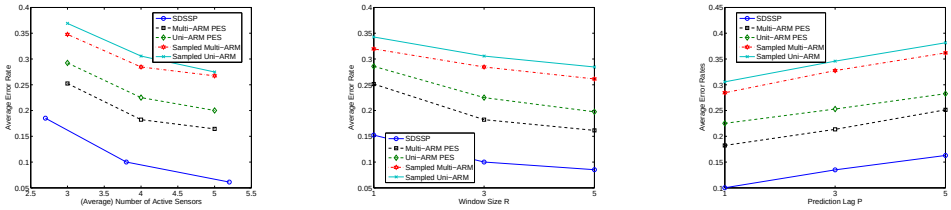
In the aforementioned experiments, the number of states are set based on the performance on an independent validation set. It is also intriguing to study the change of perfor-



(a) $P = 1, R = 1$ (b) $P = 1, R = 2$ (c) $P = 1, R = 3$

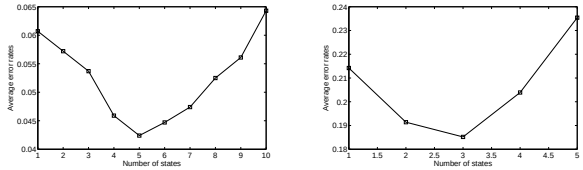
(d) $P = 2, R = 1$ (e) $P = 2, R = 2$ (f) $P = 2, R = 3$

Figure 5: Comparison between dynamic and static models for predicting P-step measurement with a window of length R. The six figures plot the average prediction errors over 33 sensors with varying P and R on the test time series. The result shows the dynamic prediction model usually has smaller prediction errors than its static counterpart most of time.



(a) Error rates (b) window size R (c) Prediction lag P

Figure 6: Comparison of error rates by the different algorithms on the ICU Sensor dataset: (a) the error rates with varying numbers of active sensors (achieved with $P = 1$ and $R = 3$); (b) the error rates with varying window sizes R when the prediction lag is fixed to $P = 1$; (c) the error rates with varying lag predictions P when window size is fixed to $R = 3$.



(a) Oil drilling rig (b) ICU

Figure 9: Change of average error rates versus different number of states.

mance when the number of states varies (*cf.* Figure 9). To ensure fair comparison between different numbers of states, the average number of active sensors is set to 8 on the oil drilling dataset and to 3 on ICU sensor dataset. From the result, we observe that neither too many nor too few states is adequate to perform well. The model with too few states underestimates the system complexity, making it incapable of capturing the conditions of a rapid-changing system. On the other hand, a model with too many states tends to overfit with the sensor time-series data.

7. CONCLUSION

This paper presents a state-driven dynamic sensor selection and prediction model in which a subset of active sensors are dynamically selected over time. A state-stacked sparseness algorithm is developed to select the active sensors by minimizing the prediction error over time in a state-driven dynamic system. We derive an efficient algorithm which converges at a rate of $O(1/L^2)$ to an optimal sensor selection and prediction model, where L is the number of iterations. We apply the algorithm to real-world applications of two different industry domains – oil drilling sensors and ICU sensors. The results show that the algorithm can achieve the best performance in terms of error rates in tracking and predicting the sensor readings over time.

8. REFERENCES

[1] C. Aggarwal, Y. Xie, and P. Yu. On dynamic data-driven selection of sensor streams. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2011.

[2] C. C. Aggarwal, editor. *Data Streams: Models and Algorithms*. Springer, 2007.

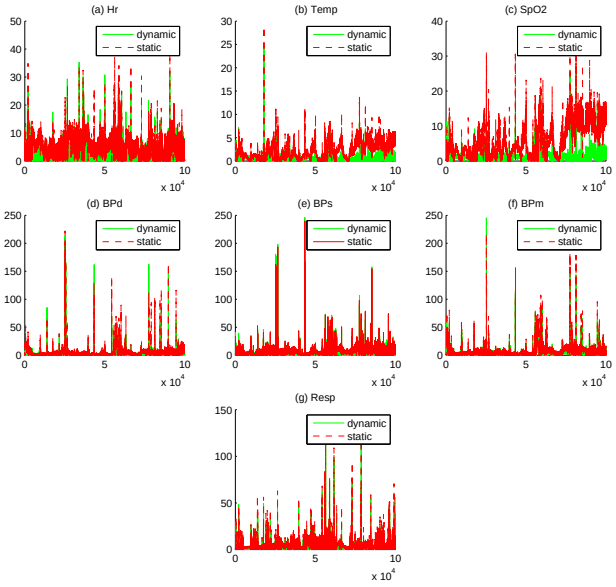


Figure 8: Comparison between dynamic and static models on seven ICU sensors over a series of $10,0000$ sensor measurements. The result is obtained with $P = 1$, and $R = 3$.

[3] C. C. Aggarwal, A. Bar-Noy, and S. Shamoun. On sensor selection in linked information networks. In *IEEE DCOSS Conference*, 2011.

[4] C. Anagnostopoulos, N. M. Adams, and D. J. Hand. Streaming covariance selection with applications to adaptive querying in sensor networks. *The Computer Journal*, 53(9):1401–1414, 2010.

[5] T. Arici, T. Akgun, and Y. Altunbasak. A prediction error-based hypothesis testing method for sensor data acquisition. *ACM Transactions on Sensor Networks*, 2:529–556, 2006.

[6] S. Chang, G.-J. Qi, C. C. Aggarwal, J. Zhou, M. Wang, and T. S. Huang. Factorized similarity learning in networks. pages 60–69, 2014.

[7] A. Deligiannakis and Y. Kotidis. Data reduction techniques in sensor networks. *IEEE Data Engineering Bulletin*, 28(1):19–25, 2005.

[8] D. Golovin, M.Faulkner, and A. Krause. Online distributed sensor selection. In *IPSN Conference*, 2010.

[9] V. Klemas. Tracking oil slicks and predicting their trajectories using remote sensors and models: case studies of the sea princess and deepwater horizon oil spills. *Journal of Coastal Research*, pages 789–797, 2010.

[10] G. Kollios, J. Byers, J. Considline, M. Hadjieleftheriou, and F. Li. Robust aggregation in sensor networks. *IEEE Data Engineering Bulletin*, 28(1):26–32, 2005.

[11] A. Markov. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. 1971.

[12] S. Nassar, K.-P. SCHWARZ, N. EL-SHEIMY, and A. Noureldin. Modeling inertial sensor errors using autoregressive (ar) models. *Navigation*, 51(4):259–268, 2004.

[13] Y. Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.

[14] S. Papadimitriou, J. Sun, and C. Faloutsos. *Data Streams: Models and Algorithms*, chapter Dimensionality Reduction and Forecasting of Time-Series Data Streams, pages 261–288. Springer, 2007.

[15] G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Online community detection in social sensing. In *WSDM*, pages 617–626. ACM, 2013.

[16] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *ACM SIGMOD Conference*, July 2005.

[17] S. Sharshar, L. Allart, and M.-C. Chambrin. A new approach to the abstraction of monitoring data in intensive care. In *Artificial Intelligence in Medicine*, pages 13–22. Springer, 2005.

[18] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM J. Optim.*, May 2008.

[19] L. Yann-Ael, S. Santini, and G. Bontempi. Adaptive model selection for time series prediction in wireless sensor networks. *Signal Processing*, 87:3010–3020, 2007.

[20] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *ICDE*, 2000.

[21] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.