OPOSSAM: Online Prediction of Stream Data with Self-Adaptive Memory

Akihiro Yamaguchi[1], Shigeru Maya[1], Tatsuya Inagi[1], Ken Ueno[1]
[1] System Engineering Lab., Corporate R&D Center, Toshiba Corporation, Japan
akihiro5.yamaguchi@toshiba.co.jp

***Abstract*—Data streams such as traffic flows, stock prices, and electricity consumption are endless time-series data from time-varying environments, and concept drift in non-stationary data streams is an important problem. To forecast the short-range future values of such data streams accurately in real time, we propose an online prediction method called Online Prediction Of Stream data with Self-Adaptive Memory (OPOSSAM). OPOSSAM introduces adaptive memory management consisting of short-term memory and long-term memory to manage time-series segments, and forecasts future values by local regression based on similar time-series segments. In order to deal with concept drift, OPOSSAM automatically adjusts the prediction model learned from short-term memory by considering the prediction model learned from the entire memory as the prior model. In addition, OPOSSAM keeps long-term memory consistent by reducing redundant samples with large prediction errors. Experimental results showed a reduction in prediction errors compared with baseline methods on real-world datasets in the different domains of traffic flow, stock prices, and electricity consumption.**

***Keywords*-Data stream; Online prediction; Short-range forecasting; Concept drift; Biased L2-regularization**

I. Introduction

Data streams are endless non-stationary time-series data from time-varying environments, and real-time analysis of such data is the foundation of Internet of Things technology [1]. The analysis of such data streams requires (1) using a finite memory capacity to handle an infinite volume of data and (2) dealing with concept drift for non-stationary data characteristics.

Concept drift occurs in several patterns, such as sudden drift, incremental drift, and recurring drift [2], and machine learning techniques that can deal with concept drift have attracted growing interest. A sliding window, which stores the most recent part of the data stream in short-term memory, has been widely adopted to handle sudden and incremental concept drift. However, it is difficult to determine the window size for non-stationary data streams because of the following issues: (a) it is difficult to handle sudden changes when the window size is too large, and (b) stationary data streams are vulnerable to noise when the window size is too small. In addition, if similar concepts reoccur after a long period of time, the sliding window cannot utilize former concepts. To overcome these problems, Self-Adjusting Memory (SAM) has been proposed as a classification method with both Long-Term Memory (LTM) and Short-Term Memory (STM) [3], [4]. SAM uses multiple different window sizes in STM, compresses older stream data into LTM, and adaptively uses multiple sample sets obtained by combining them. This provides high classification accuracy for synthetic datasets with heterogeneous concept drift and for real-world datasets, without tuning of meta-parameters among datasets.

For data streams such as traffic flows, stock prices, and electricity consumption, online prediction methods are needed to forecast short-range future values accurately in real time. As an example of traffic flow prediction, applications such as real-time car navigation systems and advanced traffic management systems need to accurately forecast traffic flows several time steps in advance by using traffic flow data delivered at 5-min intervals [5], [6]. One approach to such prediction of future values is based on regression from recent time-series stored using a sliding window. To improve accuracy, existing methods that use non-linear regression [7] and adaptive window sizes [8] have been proposed. Another approach is based on past history and stores time-series segments in order to forecast future values using segments similar to the current segment [9]. This approach has been used to forecast traffic flows, stock prices, and electricity consumption by combining it with domain-specific approach.

However, it is difficult for existing methods such as [7], [8], [9] to accurately forecast various data streams that have heterogeneous concept drift. Existing regression-based methods using a sliding window do not utilize past information. In contrast, existing history-based methods collect the history in a fixed period before prediction, which makes it difficult to take concept drift into account. Although SAM considers concept drift and manages past information, it is aimed at classification rather than online prediction, and it is not appropriate to apply SAM to online prediction from the perspectives of maintaining LTM consistency and requiring multiple learning models based on various sample sets.

In this paper, we propose the Online Prediction method Of Stream data with Self-Adaptive Memory (OPOSSAM) in order to accurately forecast short-range future values from various data streams under heterogeneous concept drift. Inspired by SAM [3], [4], we introduce adaptive memory management, consisting of STM and LTM. to online prediction. The memory manages time-series segments from the data stream, and the local regression of similar time-series segments is applied based on [9]. OPOSSAM keeps LTM

consistent by reducing redundant samples with large prediction errors. Furthermore, the regularization-based adaptation of OPOSSAM adjusts the prediction model learned from STM by considering the prediction model learned from the entire memory as the prior model. The approach not only avoids overtraining from STM, which may not store enough samples, but also adjusts the importance between STM and LTM based on concept drift. In addition, because OPOSSAM precomputes the prediction model, adjustments can be calculated efficiently without the need to learn many models from scratch.

The main contributions are summarized as follows:

- We propose OPOSSAM, which
  - introduces adaptive memory management consisting of STM and LTM to online prediction;
  - keeps LTM consistent and redundant in a way appropriate to online prediction based on local regression; and
  - uses regularization-based adaptation to adjust the importance between STM and LTM in order to precompute the prediction model.
- We confirm superiority in terms of accuracy by
  - demonstrating that the mechanism of OPOSSAM is effective by using simple synthetic datasets; and
  - reducing prediction errors compared with baseline methods on real-world datasets in the different domains of traffic flow, stock prices, and electricity consumption.

This paper is an extended version of our previous work [10], and is organized as follows. Section II provides an overview of related work. In Section III, we introduce the problem setting. In Section IV, we propose OPOSSAM. Section V shows the experiments, and our conclusions are provided in Section VI.

II. Related Work

Regression-based predictors using a sliding window of fixed size are widely used in online prediction methods. Although linear predictors such as AR are well known, their accuracy tends to be low for non-linear real-world data. To improve accuracy, non-linear regression-based methods such as support vector regression, kernel ridge regression, and neural networks have been proposed [6], [7], [11]. Learning of non-linear regression-based predictors is time-consuming for real-time forecasting in some cases. To solve this problem, a predictor that learns the kernel ridge regression in an incremental fashion by employing a sliding window has been proposed [7]. However, these methods discard old information from the sliding window. In addition, it is difficult to satisfy both robustness against noise and adaptation to sudden drift when using a fixed window size.

History-based predictors have also been proposed in order to improve accuracy. Although domain-specific knowledge such as large traffic flows during rush hour can be well utilized, this is not generalizable [5], [9]. Another approach stores past time-series segments and forecasts future values based on previous segments that are similar to the current segment. This approach is widely used in several domains, such as traffic flows, stock prices, and electricity consumption, and achieves high accuracy in each domain [9], [12], [13]. However, the existing methods based on this approach store segments in fixed intervals before the start of forecasting, and always use them during prediction. As a result, the old information is always used even when concept drift occurs. The obvious solution is to manage time-series segments from the data stream by using a sliding window of fixed size. However, old information is discarded once it has moved out of the sliding window.

Online prediction methods dealing with concept drift have been proposed. In [14], [15], [16], [17], regression based on decision tree models was proposed for data streams. However, these methods do not utilize LTM. In addition, they assume the case of forecasting values one step ahead based on multivariate data streams, which is appropriate when the number of variables is large. In contrast, we focus on data streams where the number of variables is small (typically univariate streams) and forecasting N step ahead where N is a small number. The method in [18] combines prediction with a multiple time-scale structure, and selects a predictor that fits the current environment for each time-scale. However, this is aimed at long-range forecasts, not short-range forecasts. In addition, predictors are generated in real time, and the number of predictors may increase without bound. In contrast, an online prediction method that adjusts the sliding window size without explicitly detecting concept drift has been proposed [8]. This delivers higher accuracy than methods based on [19]. However, since it uses only STM and discards older information, it cannot utilize LTM. In addition, the predictor is linear, and the accuracy has not been compared with that of non-linear methods.

Recently, SAM [3], [4] has been proposed as a classification method based on the k-nearest neighbor algorithm. SAM utilizes multiple different-sized window in STM like [8], compresses older information in LTM for recurring drift, and combines them dynamically. However, application in forecasting is difficult for the following reasons.

- Although SAM judges the consistency of LTM based on class labels, there are no labels in the case of online prediction. In addition, because observed values may be anomalous and/or noisy in real-world online prediction, we cannot always trust the values, unlike class labels.
- SAM needs various learning models for multiple sample sets from different window sizes and from the combination of STM and LTM. For this reason, if we apply it to local regression based on the k-nearest neighbor method directly, it needs to learn regression models many times, which increases the computational

cost.

Domain adaptation and transfer learning have also recently become attractive research topics. The objective is to learn a target domain model with few samples by utilizing the knowledge of a source domain for which there are many samples. The idea of a prior model is to use the source model as a prior for the model parameters of a target model that is trained on the target data. In the case of Regularized Least Squares (RLS), a regularization term on the model parameters $\mathbf{w}$ of the form $||\mathbf{w}||^2$ is simply replaced with the biased regularization term $||\mathbf{w} - \mathbf{w}'||^2$, where $\mathbf{w}'$ is the model parameters learned in the source model. The idea has been successful both practically and theoretically [20], [21], [22], [23]. In addition, the computational cost is low. In [24], transfer learning was extended to a temporal representation and used for learning under concept drift. However, its aim was classification. In addition, the idea is not regularization-based, and adds the features of the source domain into those of the target domain. As a result, because the dimension of the feature space increases, learning algorithms such as the k-nearest neighbor algorithm suffer from problems with dimensionality.

III. Problem Setting

This section describes the problem setting from the three perspectives of online prediction, concept drift, and memory management.

A. Online prediction on data streams

A data stream is a sequence $(\mathbf{x}_1, \mathbf{x}_2, \cdots)$ of tuples where each tuple $\mathbf{x}_i$ is a J-dimensional vector $(\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \cdots, \mathbf{x}_{i,J})$. At the current time step t, we seek to forecast value $\hat{\mathbf{x}}_{t+N,j}$ of the jth variable for N steps ahead. The number of prediction steps N, tuple dimension J, and predicted variable j are given as constant small numbers before prediction. We focus on (but are not limited to) univariate data streams (i.e., $J = 1$) and denote $\mathbf{x}'_i = \mathbf{x}_{i,1}$ and $\hat{\mathbf{x}}'_i = \hat{\mathbf{x}}_{i,1}$. In addition, we assume short-range forecasting (typically $N = 1, 2, \cdots, 5$). We denote $y_{i+N} = \mathbf{x}_{i+N,j}$ and $\hat{y}_{i+N} = \hat{\mathbf{x}}_{i+N,j}$.

After $\hat{y}_t$ is predicted at time step $t-N$, the observed value y_t is given at time step t. A predictor can then measure the prediction error, which is the error between the predicted and observed values, given by

$$\mathrm{loss}(\hat{y}_t, y_t) = (\hat{y}_t - y_t)^2. \tag{1}$$

B. Concept drift

Concept drift [2] occurs when the joint distribution changes for at least two time steps t_i and t_j:

$$P(\mathbf{x}_{t_i-D'+1}, \mathbf{x}_{t_i-D'+2}, \cdots, \mathbf{x}_{t_i}, y_{t_i+N}) \neq P(\mathbf{x}_{t_j-D'+1}, \mathbf{x}_{t_j-D'+2}, \cdots, \mathbf{x}_{t_j}, y_{t_j+N}) \tag{2}$$

where D' is a certain number. The joint distribution can also be written as

$$P(\mathbf{x}_{t-D'+1}, \cdots, \mathbf{x}_t, y_{t+N}) = P(\mathbf{x}_{t-D'+1}, \cdots, \mathbf{x}_t) P(y_{t+N} \mid \mathbf{x}_{t-D'+1}, \cdots, \mathbf{x}_t) \tag{3}$$

where $P(\mathbf{x}_{t-D'+1}, \cdots, \mathbf{x}_t)$ is the distribution of the features and $P(y_{t+N} \mid \mathbf{x}_{t-D'+1}, \cdots, \mathbf{x}_t)$ is the posterior probability of the prediction. When $P(y_{t+N} \mid \mathbf{x}_{t-D'+1}, \cdots, \mathbf{x}_t)$ changes over time, it is called real drift. When $P(\mathbf{x}_{t-D'+1}, \cdots, \mathbf{x}_t)$ changes without affecting $P(y_{t+N} \mid \mathbf{x}_{t-D'+1}, \cdots, \mathbf{x}_t)$, it is called virtual drift. Another point of view is that when the distribution changes quickly and severely, it is called sudden drift. In contrast, when the distribution evolves slowly, it is called incremental drift. In addition, when changes in the distribution occur repeatedly, it is called recurring drift.

C. Memory management

A time-series segment $\mathbf{s}_t$ of length D at time step t is represented as a DJ-dimensional vector $(\mathbf{x}_{t-D+1}, \cdots, \mathbf{x}_t)$. The memory of the proposed method stores each $DJ + 1$-dimensional vector $\mathbf{z}_i = (\mathbf{s}_i, y_{i+N})$ as a sample. Memory capacity and length D are given before prediction ($D = 5$ as default). We discuss the memory capacity by using the maximum number of samples that can be stored in the entire memory, and denote the number as $L_{\max}$.

IV. Proposed Method

In this section, we give an overview of the proposed method, OPOSSAM, and then describe each of its components.

A. Overview

Figure 1 shows an overview of OPOSSAM. Inspired by the idea of SAM, OPOSSAM manages STM and LTM separately in order to tackle the following issues: (a) STM cannot handle recurring old information well but does deal with sudden drift; and (b) LTM cannot handle sudden drift well but does deal with recurring drift. STM manages only recent samples using a sliding window. LTM manages older samples transferred from STM by choosing redundant samples in dense regions and removing anomalous and/or noisy samples with the largest prediction errors to keep the number of samples below $L_{\max}$. The memory management of STM and LTM is described in Section IV-C and Section IV-D, respectively.

First, given the current time-series segment from the data stream and the samples of $(\mathbf{s}_i, y_{i+N})$ from the entire memory consisting of STM and LTM, OPOSSAM executes the following local regression of similar time-series segments:

(1) Find K time-series segments $\mathbf{s}_{i_1}, \mathbf{s}_{i_2}, \cdots, \mathbf{s}_{i_K}$ similar to $\mathbf{s}_{t_1}$ at the current time t_1 from the entire memory consisting of STM and LTM;

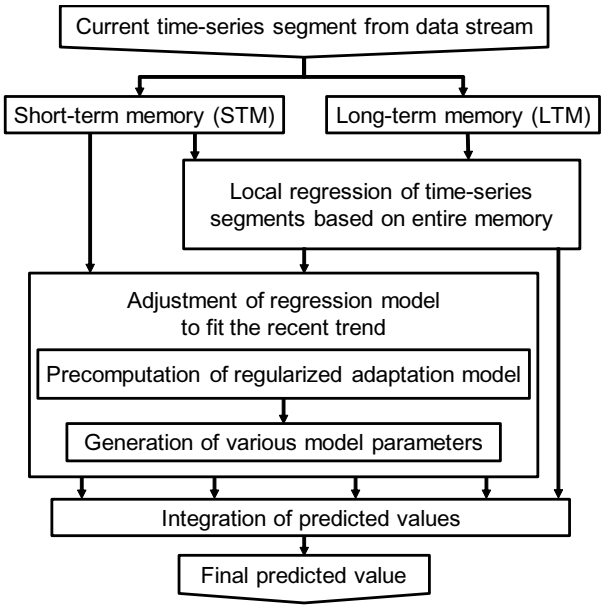

Figure 1. Overview of OPOSSAM

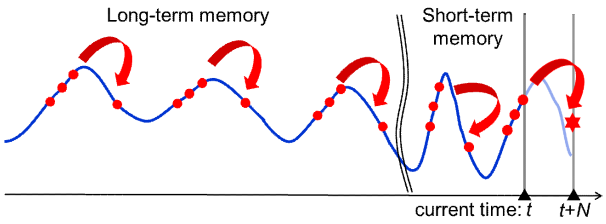

Figure 2. Illustration of local regression of time-series segments based on the entire memory in the case of $D = 3$ and $J = 1$ at current time step t.

(2) Derive the regression equation that estimates $y_{i_1+N}, y_{i_2+N}, \cdots, y_{i_K+N}$ from $\mathbf{s}_{i_1}, \mathbf{s}_{i_2}, \cdots, \mathbf{s}_{i_K}$; and

(3) Predict forecasting value $\hat{y}_{t_1+N}$ for N steps ahead by substituting the current time-series segment $\mathbf{s}_{t_1}$ into the regression equation.

Figure 2 illustrates a case where the length of each univariate time-series segment is 3 (i.e., $D = 3$) and the number of similar time-series segments is 4 (i.e., $K = 4$). By using the above local regression based on lazy learning, OPOSSAM is expected to deal not only with non-linear data streams but also with concept drift quickly and flexibly [25]. The formulation is given in Section IV-B.

Next, in order to adjust the influence of recent information, OPOSSAM adapts the above regression model learned from the entire memory to small samples of STM. Figure 3 illustrates a case where the length of each univariate time-series segment is 3 (i.e., $D = 3$). This is implemented in OPOSSAM by using regularization-based adaptation, which is known as a successful method in the field of domain

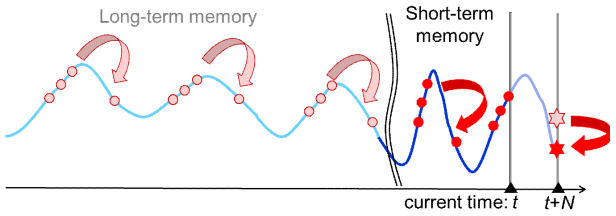

Figure 3. Illustration of adjustment of regression model so as to fit the recent tendency in the case of $D = 3$ and $J = 1$ at current time step t.

adaptation and transfer learning, as follows:

(1) Define a prior model by regression model parameters $\tilde{\mathbf{w}}$ that were learned from samples in the entire memory consisting of LTM and STM; and

(2) Learn the regression model parameters $\mathbf{w}$ with biased regularization term $\lambda||\mathbf{w} - \tilde{\mathbf{w}}||^2$ from samples of STM where λ is the regularization parameter.

The regularization parameter λ represents the importance of STM and LTM. OPOSSAM generates candidates for model parameters $\mathbf{w}$ by dynamically varying the values of λ. We show that parts with high computational complexity can be precomputed independently of λ, as described later in Section IV-E. This precomputation allows OPOSSAM to efficiently learn regression model parameters $\mathbf{w}$ adapted for various values of λ. We describe the details in Section IV-E.

Finally, OPOSSAM mixes the predicted values of the regularized adaptation model with various regularization parameter values, and selects the final predicted value using the mixed predicted value and the predicted value based on the entire memory. OPOSSAM can handle concept drift without tuning meta-parameters because it automatically adjusts the integration based on recent prediction errors. We describe the details in Section IV-F.

B. Local regression of similar time-series segments

Given samples of $(\mathbf{s}_i, y_{i+N})$ and time-series segment $\mathbf{s}_{t_1}$ at current time step t_1, we formulate the forecasting method. The idea is the same as in the existing method [9] excluding domain-specific parts. We call this method Local Regression of Similar Time-series Segments (LRSTS).

By using the Euclidean distance between $\mathbf{s}_{t_1}$ and $\mathbf{s}_i$ (i.e., $||\mathbf{s}_{t_1} - \mathbf{s}_i||^2$), we find the K nearest neighbors $\mathbf{s}_{i_1}, \mathbf{s}_{i_2}, \cdots, \mathbf{s}_{i_K}$ for $\mathbf{s}_{t_1}$ from the samples. Instead of the number of nearest neighbors K, we determine the ratio of nearest neighbors r over the samples. Namely, K is equal to $I \times r$ where I is the number of samples. It is more robust in the feature space of $\mathbf{s}_i$ to determine r instead of K because the region of nearest neighbors is less dependent on the number of samples, as suggested in [26]. We set $r = 0.1$ as the default as in [26]. OPOSSAM can search K-nearest neighbors efficiently by indexing such as KD-tree [27].

We estimate the model parameter $\mathbf{w}=(\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_D)^{\mathrm{T}}$ of linear ridge regression in

$$\min_{\mathbf{w}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 + \lambda' ||\mathbf{w}||^2,$$
$$\mathbf{X} = \begin{bmatrix} \mathbf{s}_{i_1} \\ \mathbf{s}_{i_2} \\ \vdots \\ \mathbf{s}_{i_K} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_{i_1+N} \\ y_{i_2+N} \\ \vdots \\ y_{i_K+N} \end{bmatrix} \tag{4}$$

where $\mathbf{X}$ and $\mathbf{y}$ are $K \times D$ and $K \times 1$ matrices, respectively. The optimum model parameter $\mathbf{w}$ of Eq. (4) can be expressed from the first-order optimality condition in closed form

$$\mathbf{w} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda'\mathbf{I}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{5}$$

where $\mathbf{I}$ is the $D \times D$ identity matrix. In this study, we set the regularization parameter in Eq. (4) and Eq. (5) to 0 (i.e., $\lambda' = 0$) in the same manner as the existing method [9] for simplicity of discussion. Predicted value $\hat{y}_{t_1+N}$ is calculated using

$$\hat{y}_{t_1+N} = \mathbf{s}_{t_1}\mathbf{w}. \tag{6}$$

C. STM: Short-term memory

Because recent samples have fresh information and are often more valuable than old samples for forecasting data streams with concept drift, STM stores recent samples using a sliding window. Long windows cannot handle sudden drift whereas short windows are susceptible to noise. To tackle this issue, SAM prepares learning models of various window sizes. In contrast, OPOSSAM uses regularized adaptation, which is described later in Section IV-E. For this reason, OPOSSAM manages STM with only a small window for storing recent samples.

When a tuple $\mathbf{x}_{t_1}$ is observed at current time t_1, $\mathbf{z}_{t_0} = (\mathbf{s}_{t_0}, \mathbf{x}_{t_0+N,j})$ is added to the STM as the newest sample where $t_0 = t_1 - N$. The sliding window stores recent samples

$$\begin{cases} \bigcup_{t=1}^{t_0} \mathbf{z}_t & (t_0 \leq L_{\min}) \\ \bigcup_{t=t_0-L_{\min}+1}^{t_0} \mathbf{z}_t & (L_{\min} < t_0) \end{cases} \tag{7}$$

where $L_{\min}$ is the length of the window. When sample $\mathbf{z}_{t_0}$ is added to STM for the case of $L_{\min} < t_0$, the oldest sample $\mathbf{z}_{t_0-L_{\min}}$ of STM is transferred to LTM.

D. LTM: Long-term memory

Given the maximum number of samples in both LTM and STM as $L_{\max}$, LTM needs to keep the number of samples within $L_{\max} - L_{\min}$ because STM has $L_{\min}$ samples. LTM considers non-redundancy and consistency instead of freshness, and drops the following samples if the number of samples exceeds $L_{\max} - L_{\min}$:

(A) Samples from dense regions rather than isolated points for eliminating redundant information; and

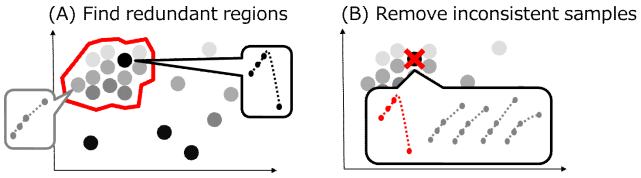

Figure 4. Illustration of how to keep samples in LTM

(B) Anomalous and/or noisy samples with large prediction errors for maintaining consistency.

Figure 4 illustrates the case where each sample is projected onto two dimensions. The procedure is as follows.

First, to find (A), OPOSSAM chooses several samples (10 samples by default) as candidates from LTM uniformly at random without replacement, and selects the sample $(\mathbf{s}_i, y_{i+N})$ of the highest density region. In order to select $(\mathbf{s}_i, y_{i+N})$, OPOSSAM finds K nearest neighbors $\mathbf{s}_{i_1}, \mathbf{s}_{i_2}, \cdots, \mathbf{s}_{i_K}$ for each candidate $\mathbf{s}_i$[1], and selects the sample that has the smallest distance from $\mathbf{s}_{i_K}$ as $\mathbf{s}_i$.

Next, to find (B), OPOSSAM applies LRSTS, described in Section IV-B, to the sample $(\mathbf{s}_i, y_{i+N})$, and derives the model parameters by Eq. (5). After this, it calculates the prediction errors $\mathrm{loss}(\hat{y}_{i_1}, y_{i_1}), \mathrm{loss}(\hat{y}_{i_2}, y_{i_2}), \cdots, \mathrm{loss}(\hat{y}_{i_K}, y_{i_K})$ respectively, and selects the sample with the largest prediction error to drop from LTM as an inconsistent sample.

E. Regularization-based adaptation

LRSTS based on all samples in the entire memory can perform prediction with small variance, although it is difficult to consider the freshness of information. In contrast, LRSTS based on samples of STM can deal with recent changes quickly although overfitting may occur because of the small number of samples. To resolve this issue, OPOSSAM adapts LRSTS learned from the entire memory to LRSTS based on recent samples of STM.

Given model parameters $\tilde{\mathbf{w}}$ derived by Eq. (5) using samples of the entire memory and time-series segment $\mathbf{s}_{t_1}$ at current time step t_1, OPOSSAM derives model parameter $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_D)^{\mathrm{T}}$, which is adapted to STM. We can regard this approach as domain adaptation because the regression model based on the entire memory can be regarded as the source hypothesis (i.e., regularization parameter $\tilde{\mathbf{w}}$ as a prior model) and the regression model based on STM can be regarded as the target hypothesis. Analogously, we can view the formulation as a Bayesian linear regression with a $\tilde{\mathbf{w}}$-mean Gaussian prior distribution.

After we find the K nearest neighbors $\mathbf{s}_{i_1}, \mathbf{s}_{i_2}, \cdots, \mathbf{s}_{i_K}$ for $\mathbf{s}_{t_1}$ from the samples of STM[2], the formulation is based

[1]As described in Section IV-B, K is set to $I_{\mathrm{LTM}} \times r$ where I_{LTM} is the number of samples in LTM and r is the ratio of nearest neighbors.

[2]As described in Section IV-B, K is set to $I_{\mathrm{STM}} \times r$ where I_{STM} is the number of samples in STM and r is the ratio of nearest neighbors.

on biased-regularized least squares

$$\min_{\mathbf{w}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 + \lambda||\mathbf{w} - \tilde{\mathbf{w}}||^2,$$
$$\mathbf{X} = \begin{bmatrix} \mathbf{s}_{i_1} \\ \mathbf{s}_{i_2} \\ \vdots \\ \mathbf{s}_{i_K} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_{i_1+N} \\ y_{i_2+N} \\ \vdots \\ y_{i_K+N} \end{bmatrix} \tag{8}$$

where λ is the regularization parameter. The optimum model parameter $\mathbf{w}$ of Eq. (8) can be expressed from the first-order optimality condition in closed form

$$\mathbf{w} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{X}^{\mathrm{T}}\tilde{\mathbf{w}}\right) + \tilde{\mathbf{w}}. \tag{9}$$

If the value of the regularization parameter λ is close to zero, then LTM tends to be ignored and the regression model is close to the LRSTS learned from samples of STM. In contrast, if the value of λ is large, then STM tends to be ignored and the regression model is close to the LRSTS learned from samples from the entire memory. To adjust the relative importance between STM and LTM, OPOSSAM learns the regression models by using various values of λ. However, Eq. (9) includes an inverse matrix, and the calculation cost is high if we calculate it according to λ values from scratch. To reduce the calculation cost, we were inspired by [28], and consider the singular value decomposition of $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ where $\mathbf{U}$ is a $K \times K$ orthogonal matrix, $\mathbf{D}$ is a $K \times D$ diagonal matrix, and $\mathbf{V}$ is a $D \times D$ orthogonal matrix. We then re-express the model parameter $\mathbf{w}$ as

$$\mathbf{w} = \mathbf{V}\left(\mathbf{D}^2 + \lambda\mathbf{I}\right)^{-1}\mathbf{D}\mathbf{U}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{X}^{\mathrm{T}}\tilde{\mathbf{w}}\right) + \tilde{\mathbf{w}}. \tag{10}$$

Furthermore, we decompose Eq. (10) for each row as

$$\mathbf{w}_j = \frac{d_j}{d_j^2 + \lambda}\mathbf{V'}_j\mathbf{U}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{X}^{\mathrm{T}}\tilde{\mathbf{w}}\right) + \tilde{\mathbf{w}}_j,$$
$$\mathbf{V'}_j = \left[\underbrace{\mathbf{V}_j}_{D}, \underbrace{0, 0, \cdots, 0}_{K-D}\right] \tag{11}$$

where d_j is the singular value (i.e., the jth diagonal element of $\mathbf{D}$ in descending order) and $\mathbf{V'}_j$ is $1 \times K$ matrix where the jth row of $\mathbf{V}$ with $K-D$ zero padding. Because OPOSSAM precomputes d_j and $\mathbf{V'}_j\mathbf{U}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{X}^{\mathrm{T}}\tilde{\mathbf{w}}\right) + \tilde{\mathbf{w}}$, it can avoid recomputing the inverse matrix for various values of λ many times.

F. Integration of predicted values

As described in Section IV-E, since the regularized adaptation model outputs multiple predicted values based on various values of the regularization parameter, OPOSSAM needs to mix them appropriately based on recent trends. In addition, OPOSSAM needs to select the final predicted value from the value predicted by the regularized adaptation model and the value predicted by LRSTS based on the entire memory. In the following, we denote the set of regularization parameter values (i.e., variation of λ in Eq. (11)) as Λ_t at time step t.

First, OPOSSAM mixes the predicted results based on various regularization parameter values by weighting them according to the current prediction error. For $\lambda \in \Lambda_{t_1}$ at current time step t_1, we denote the predicted value of the regularized adaptation model as $\hat{y}_{t_1}^{(\lambda)}$. The weight $\mathbf{a}_\lambda$ is defined as

$$\mathbf{a}_\lambda = 1 - \widetilde{\mathrm{loss}}(\hat{y}_{t_1}^{(\lambda)}, y_{t_1}), \quad \lambda \in \Lambda_{t_1}, \tag{12}$$

where $\widetilde{\mathrm{loss}}(\cdot^{(\lambda)}, \cdot)$ rescales the range of $\{\mathrm{loss}(\cdot^{(\lambda)}, \cdot)\}_{\lambda \in \Lambda_{t_1}}$ to $[0, 1]$ linearly. The mixed predicted value $\hat{y}_{t_1+N}^{(\Lambda_{t_1})}$ at time $t_1 + N$ is then adjusted to

$$\hat{y}_{t_1+N}^{(\Lambda_{t_1})} = \sum_{\lambda \in \Lambda_{t_1}} \mathbf{a}_\lambda \hat{y}_{t_1+N}^{(\lambda)} \Big/ \sum_{\lambda \in \Lambda_{t_1}} \mathbf{a}_\lambda \tag{13}$$

The variation of λ is determined iteratively by the following procedure:

(1) calculate center m as $m := \sum_{\lambda \in \Lambda_t}(\mathbf{a}_\lambda \log_{10} \lambda)$ from current variation Λ_t; and
(2) generate the next variation Λ_{t+1} as

$$10^{m-C}, 10^{m-C+1}, \cdots, 10^{m+C}$$

where C is a constant number ($C = 10$ as default).

The initial variation Λ_1 is generated in $m = 0$ as

$$10^{-C}, 10^{-C+1}, \cdots, 10^{C}.$$

Finally, OPOSSAM selects the final predicted value between $\hat{y}_{t_1+N}^{(\Lambda_{t_1})}$, which is the mixed predicted value in Eq. (13), and $\hat{y}_{t_1+N}^{(\mathrm{all})}$, which is the predicted value of LRSTS based on the entire memory, according to the predicted errors averaged over the recent $L_{\min}$ samples:

$$\hat{y}_{t_1+N} = \begin{cases} \hat{y}_{t_1+N}^{(\Lambda_{t_1})} & \left(0 \leq \sum_{t=t_1-L_{\min}+1}^{t_1} \mathrm{loss}(\hat{y}_t^{(\mathrm{all})}, y_t) - \mathrm{loss}(\hat{y}_t^{(\Lambda_t)}, y_t)\right) \\ \hat{y}_{t_1+N}^{(\mathrm{all})} & (\text{otherwise})\,. \end{cases} \tag{14}$$

V. Experiments

In this section, we first show the baseline methods and then confirm how the proposed mechanism affects the prediction using simple synthetic toy data. After that, we compare the accuracy of the proposed method with that of the baseline methods on three real-world datasets in the domains of traffic flow, stock prices, and electricity consumption.

A. Baseline methods

We select the following baseline methods, which are appropriate for forecasting short-range future values on univariate data streams, and we compare these methods with the proposed method **OPOSSAM**. For memory capacity conditions, we set the minimum and maximum numbers of samples to store in memory at 300 and 1000, respectively (i.e., $L_{\min} = 300$ and $L_{\max} = 1000$). Unless otherwise stated, the parameter settings of the baseline methods are the same as those of OPOSSAM.

ARWin is an existing method [8] based on linear regression with adaptive window sizes. Following the authors' suggestions, we set the variation of window sizes to $3, 4, \cdots, 50$. This is an existing linear online prediction method that can handle concept drift.

KRWin is an existing method [7] based on kernel ridge regression with a sliding window of fixed size. We optimize the meta-parameters using 1000–2000 steps for each dataset. We search the window sizes in $\{300, 1000\}$ according to $L_{\min}$ and $L_{\max}$[3], and use radial basis function kernel

$$\mathrm{kernel}(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2). \quad (15)$$

We search for the value of γ such that the median of $\{\gamma ||\mathbf{x}_t - \mathbf{x}_{t'}||^2\}_{t,t'=1000}^{2000}$ is in $\{10^i\}_{i=-2,-1,\cdots,2}$ based on the idea of [29], and search the regularization parameter in $\{10^i\}_{i=-2,-1,\cdots,2}$. This is an existing non-linear online prediction method with a sliding window.

NRWin is an existing method excluding the domain-specific parts in [9], and is the same as LRSTS with a fixed size sliding window. We prepare two windows which sizes are 300 and 1000 according to $L_{\min}$ and $L_{\max}$, and denote them as NRWin300 and NRWin1000 respectively. This is an existing online prediction method based on LRSTS.

NAWin is an existing method excluding the domain-specific parts in [12], [13], and uses the mean among values for N steps ahead among similar time-series segments as a predicted value. Other settings are the same as for NRWin. This is an online prediction method based on similar time-series segments.

B. Insight into mechanism of OPOSSAM on synthetic data

In this section, we confirm the effectiveness of the proposed mechanism by visualizing the prediction results using three simple synthetic data streams. Each data stream is univariate (i.e., $J = 1$). In each data stream, we add a small amount of noise based on a normal distribution with mean zero and standard deviation 0.01, and evaluate the forecast for 3 steps ahead (i.e., $N = 3$).

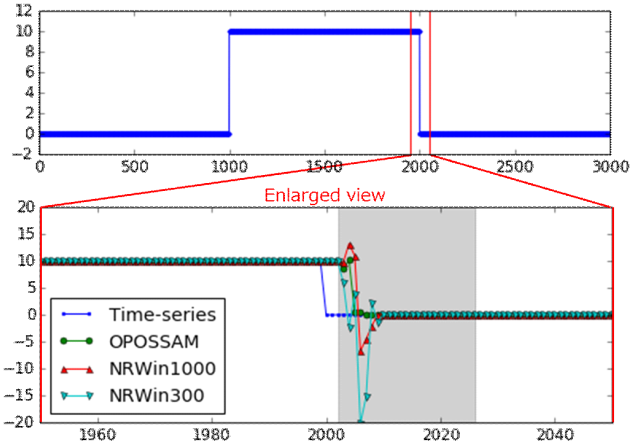

Figure 5. Effect of introducing LTM (top: synthetic data; bottom: closeup of the most different part)

Evaluation 1: Effect of introducing LTM

We confirm the effectiveness of introducing LTM in addition to STM. We use the synthetic data in Fig. 5 (top) which contains sudden concept drift at steps 1000 and 2000, and also includes recurring concept drift between steps 0–1000 and 2000–3000. We compare the prediction results of OPOSSAM with NRWin1000 and NRWin300 because they use LRSTS, which OPOSSAM also uses, and NRWin1000 has competitive accuracy as described later in Section V-C.

Figure 5 (bottom) focuses on the most different part (range surrounded by red vertical lines in the top of Fig. 5) and shows the prediction results. We find that, from when the second sudden concept drift occurs to several steps later, all of the methods have difficulty in forecasting. However, the deviation is smaller for OPOSSAM than for the comparison methods. The gray region in Fig. 5 (bottom) shows that OPOSSAM selected $\hat{y}_{t_1+N}^{(\mathrm{all})}$ instead of $\hat{y}_{t_1+N}^{(\Lambda_{t_1})}$ in Eq. (14). This indicates that OPOSSAM does not make STM dominant in the gray region. In other words, OPOSSAM preferentially uses LTM for several steps after the second sudden concept drift occurs.

The above observation shows that OPOSSAM can utilize the old information that appeared during steps 0–1000 when recurring concept drift occurs from step 2000. In contrast, the other methods discarded the old information because they do not have LTM. Therefore, we find that OPOSSAM utilizes LTM effectively in the example.

Evaluation 2: Effect of regularization-based adjustment

We confirm the effectiveness of dynamically changing the weights of the regularization parameter values in the

[3]Because KRWin needs to store the gram matrix and the inverse matrix, the actual memory capacity that is required is on the order of the square of the number of samples [7]. However, we search the window sizes by matching the numbers of samples.

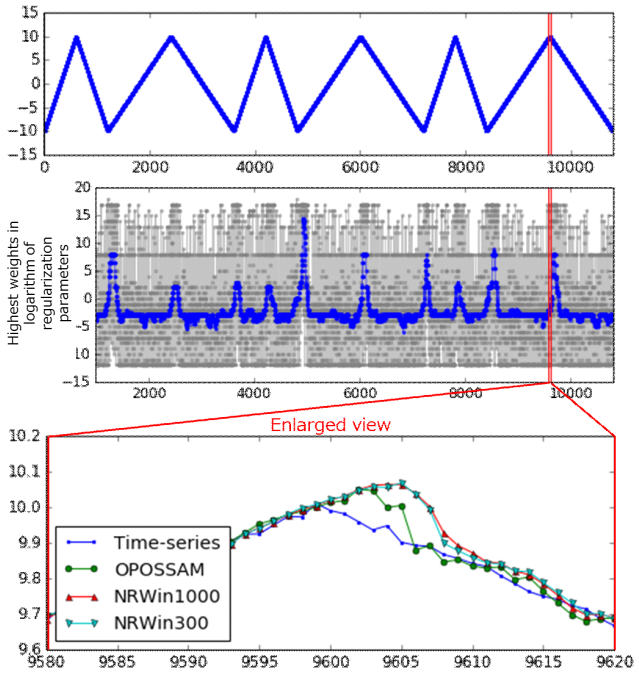

Figure 6. Effect of regularization-based adjustment (top: synthetic data; middle: weights of regularization parameters; bottom: closeup of one of the most different parts)

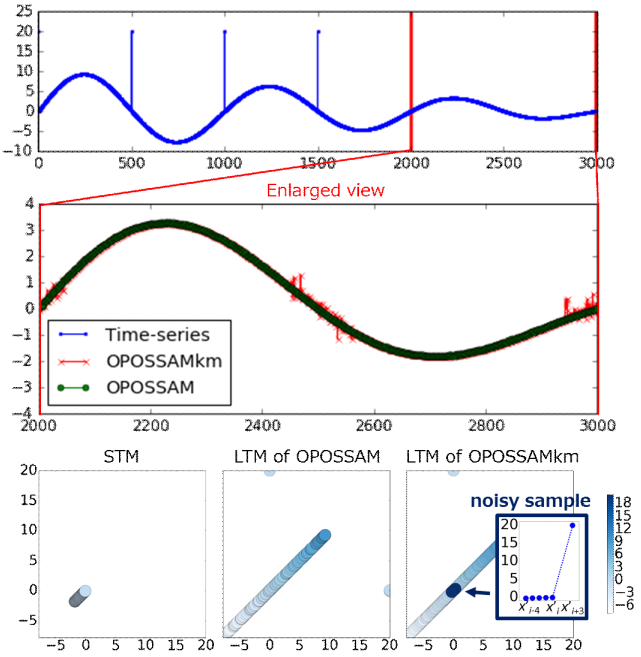

Figure 7. Effect of maintaining consistency in LTM (top: synthetic data; middle: closeup of the most different part; bottom: projected samples in memory)

regularized adaptation model. We use the synthetic data in Fig. 6 (top), which consists of multiple connected straight lines with recurring concept drift. As in the previous experiment, we compare the prediction results of OPOSSAM with NRWin1000 and NRWin300. Unlike the previous experiment, OPOSSAM always selected $\hat{y}_{t_1+N}^{(\Lambda_{t_1})}$ in Eq. (14) in this experiment.

Figure 6 (middle) shows changes in the weights of the regularization parameters. The vertical axis is the logarithm of the regularization parameter (i.e., $\log_{10}\lambda$). Although each gray plot represents a regularization parameter λ of the highest weight $\mathbf{a}_\lambda$ in Eq. (12), the changes fluctuate and it is difficult to find a trend despite the simplicity of the data. For this reason, we apply a median filter of window size 100 and show the results as blue plots. This shows that the regularization parameters tend to be adjusted to high values several steps after the straight line bends. This is because it is difficult to forecast this kind of step from only the recent data of STM and OPOSSAM increases the importance of LTM to utilize old similar information.

For several steps after a straight line bends, forecasting is difficult and the prediction results differ most significantly among the methods. Figure 6 (bottom) focuses on one of the parts (range surrounded by red vertical lines in the top and middle of Fig. 6) and shows the prediction results. We find that deviation is smaller for OPOSSAM than for the comparison methods. This is because OPOSSAM adjusts the regularization parameter to make LTM dominant during steps where it is difficult to forecast using only STM.

The above observation shows that OPOSSAM can adjust the weights of the regularization parameter values in the regularized adaptation model effectively in this example, although the weights fluctuate without using a median filter.

Evaluation 3: Effect of maintaining consistency in LTM

We checked the effectiveness of the mechanism for maintaining consistency in LTM by using the synthetic data in Fig. 7 (top). These data are a sine curve that decreases linearly in amplitude from 10 to 1, and includes incremental and recurring concept drift. In addition, we add three anomalies where $y_t = 20$ at each step $t = 500, 1000, 1500$. We modify OPOSSAM so that the samples are compressed as centroids by k-means++ clustering in the same way as SAM [3], [4] instead of the mechanism described in Section IV-D. We denote the modified version as **OPOSSAMkm**, and compare it with OPOSSAM.

Figure 7 (middle) focuses on the last 1000 steps (range surrounded by red vertical lines in the top of Fig. 7), where no anomalies are present, and shows the prediction results. We find that it is difficult for OPOSSAMkm to forecast around the steps where y_t (the value on the vertical axis in the middle of Fig. 7) is close to zero. This is because OPOSSAMkm cannot remove the anomalies from LTM and applies LRSTS to samples including the anomalies. In contrast, OPOSSAM can make a correct forecast because

it removes anomalous samples by the mechanism for maintaining consistency in LTM.

Here we confirm in detail the information discussed above. Figure 7 (bottom) shows the samples from the STM[4], LTM in OPOSSAM, and LTM in OPOSSAMkm, respectively at the last step. For each sample $\mathbf{z}_i = (\mathbf{x}'_{i-4}, \mathbf{x}'_{i-3}, \cdots, \mathbf{x}'_i, y_{i+3})$, the value of the horizontal axis is the value of $\mathbf{x}'_{i-1}$, the value of vertical axis is the value of $\mathbf{x}'_i$, and the color corresponds to the value of y_{i+3}, respectively. Although there are no anomalous samples in LTM of OPOSSAM, there are anomalous samples around $(\mathbf{x}'_{i-1}, \mathbf{x}'_i) = (0, 0)$ (dark blue points in the bottom of Fig. 7) in LTM of OPOSSAMkm.

The above observation shows that OPOSSAM can maintain consistency by keeping not only old but also important samples in LTM. In contrast, the existing mechanism [3], [4] based on k-means++ clustering cannot maintain consistency although it can keep old samples. Through the above evaluations 1–3, we find that the mechanism of OPOSSAM is effective for forecasting in those examples.

C. Comparison results of accuracy on real-world datasets

1) Real-world datasets: We use the following real-world data in each domain as univariate data streams, and remove missing values beforehand.

Traffic is the average traffic speed on a freeway operated by the California Department of Transportation[5]. It is widely used in traffic flow prediction [6], [9]. We use data observed at the traffic detector VDS:407750 from 1 October 2017 to 2 December 2017. Each step interval is 5 min. We use only the Lane 1 Speed (mph) column. The total number of steps is 18,143, and there are no missing values.

Stock is the closing price of 225 Japanese representative companies listed on the Tokyo Stock Exchange[6]. It was used to evaluate ARWin in [8]. In the same way as [8], we use the Close column only. The period is from 19 May 1997 to 15 May 2017. Each step interval is 1 day. The total number of steps is 5046 including 143 missing values.

Electricity is the electric consumption for a single residential customer in France[7] [30]. The consumption measurements were gathered between December 2006 and November 2010 with 1 min resolution. We use only the Voltage column. Each step interval is 1 min. The total number of steps is 2,075,259 including 25,979 missing values.

2) Experimental setup: We measure accuracy using the three metrics of Root Mean Squared Error (RMSE), Mean Absolute Error (RAE), and Median Absolute Error (MdAE). The metrics are measured in an interleaved test-then-train or prequential manner, which is standard for evaluation of data streams with concept drift. To our knowledge, this is the first time LTM has been used for online prediction of data streams with concept drift, and we focus on evaluation after LTM has stored enough samples. For this reason, we skip the first 1000 steps and evaluate the remaining steps in each dataset. We evaluate the forecasting results for 1, 3, and 5 steps ahead (i.e., $N = 1, 3, 5$) as prediction steps. Because OPOSSAM involves a random sampling process, the results may vary between different runs, and we repeat the experiments 10 times on each dataset and state the mean of the results.

3) Experimental results: As shown in Table I, OPOSSAM achieves the best performance in most cases. OPOSSAM has the best average ranks of 1.66, 1.18, and 1.00 crossing both datasets and prediction steps for RMSE, MAE, and MdAE, respectively. The second-best method for OPOSSAM is NRWin1000, and the average ranks are 1.71, 1.82, and 2.22 crossing both datasets and prediction steps for RMSE, MAE, and MdAE, respectively. We ran the well-known Wilcoxon signed rank test against all baselines and found that all results are statistically significant at $p = 0.0000$. For this reason, we find the following for the datasets:

- LRSTS, which OPOSSAM also uses as a base predictor, is effective for forecasting short-range values if we can find appropriate window sizes; and
- OPOSSAM, which is an extended method from LRSTS to deal with concept drift, achieves superior accuracy over baseline methods.

VI. Conclusion

In order to accurately forecast short-range future values on various data streams under heterogeneous concept drift, we introduced adaptive memory management, consisting of STM and LTM, to online prediction, and proposed OPOSSAM. The predictor is based on local regression of similar time-series segments, and the memory management is inspired by SAM, which is a classification method for self-adjusted memories in both STM and LTM. OPOSSAM manages LTM by reducing redundant samples with large prediction errors to maintain consistency within the maximum memory capacity. In addition, OPOSSAM adjusts the prediction model for recent trends by regularization-based adaptation from a prior model learned using the entire memory. This regularization-based adaptation can be efficiently precomputed when adjusting various values of the regularization parameter. In the experiments, we showed superiority in accuracy. We demonstrated that the proposed mechanism is effective for simple synthetic datasets, and confirmed that OPOSSAM is statistically superior to several baseline methods in terms of accuracy on three real-world datasets: traffic flow, stock prices, and electricity consumption.

[4]The samples in STM are the same for OPOSSAM and OPOSSAMkm.

[5]http://pems.dot.ca.gov/

[6]https://finance.yahoo.com/quote/%5EN225/

[7]https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption

Table I
PREDICTION PERFORMANCE COMPARISON ON REAL-WORLD DATASETS

Dataset	Method	1 step ahead forecast			3 steps ahead forecast			5 steps ahead forecast		
		RMSE	MAE	MdAE	RMSE	MAE	MdAE	RMSE	MAE	MdAE
Traffic	OPOSSAM	1.2992 (1)	0.5788 (1)	0.2655 (1)	2.2149 (1)	0.9259 (1)	0.3755 (1)	2.7185 (1)	1.1116 (1)	0.4136 (1)
	ARWin	1.4538 (3)	0.6679 (3)	0.3017 (3)	2.6787 (3)	1.2137 (3)	0.5171 (6)	3.7245 (3)	1.6646 (5)	0.6790 (6)
	NRWin1000	1.4356 (2)	0.6150 (2)	0.2778 (2)	2.6621 (2)	1.0497 (2)	0.3902 (2)	3.7070 (2)	1.3525 (2)	0.4338 (4)
	NRWin300	2.0819 (5)	0.7439 (4)	0.3078 (4)	3.9015 (6)	1.2971 (4)	0.4328 (5)	5.0048 (7)	1.6487 (4)	0.4868 (5)
	NAWin1000	3.9302 (7)	1.4246 (7)	0.3280 (6)	4.2880 (7)	1.6206 (7)	0.4050 (4)	4.5954 (6)	1.7586 (6)	0.4320 (3)
	NAWin300	3.0232 (6)	1.0870 (6)	0.3133 (5)	3.7075 (5)	1.3632 (5)	0.3933 (3)	4.2516 (5)	1.5515 (3)	0.4233 (2)
	KWRin	1.7166 (4)	0.9047 (5)	0.4305 (7)	2.9313 (4)	1.5709 (6)	0.6705 (7)	4.0727 (4)	2.1735 (7)	0.8418 (7)
Stock	OPOSSAM	200.35 (1)	142.90 (1)	106.14 (1)	347.73 (1)	252.50 (1)	190.39 (1)	444.58 (1)	331.09 (1)	259.53 (1)
	ARWin	217.63 (3)	156.52 (3)	116.87 (4)	398.31 (3)	291.65 (3)	217.14 (3)	548.54 (3)	405.62 (4)	310.29 (4)
	NRWin1000	204.90 (2)	144.48 (2)	107.11 (2)	352.06 (2)	257.52 (2)	199.34 (2)	463.63 (2)	343.89 (2)	268.67 (2)
	NRWin300	223.41 (4)	157.41 (4)	115.42 (3)	405.81 (4)	293.31 (4)	218.03 (4)	551.53 (4)	398.17 (3)	300.07 (3)
	NAWin1000	689.12 (6)	452.54 (6)	260.75 (6)	754.99 (7)	524.43 (6)	340.12 (6)	817.32 (7)	585.07 (6)	398.55 (6)
	NAWin300	413.11 (5)	279.33 (5)	190.55 (5)	515.89 (5)	370.75 (5)	276.14 (5)	601.68 (5)	442.53 (5)	338.49 (5)
	KWRin	704.42 (7)	547.84 (7)	447.71 (7)	749.19 (6)	582.47 (7)	476.09 (7)	792.83 (6)	616.26 (7)	504.68 (7)
Electricity	OPOSSAM	0.6045 (2)	0.4348 (1)	0.3176 (1)	1.2676 (6)	0.7440 (2)	0.5613 (1)	14.269 (7)	0.9110 (2)	0.6897 (1)
	ARWin	0.6936 (4)	0.5032 (4)	0.3705 (4)	1.0839 (2)	0.8035 (3)	0.6086 (3)	1.3454 (4)	1.0081 (5)	0.7750 (4)
	NRWin1000	0.6040 (1)	0.4363 (2)	0.3203 (2)	1.0032 (1)	0.7429 (1)	0.5635 (2)	1.2105 (1)	0.9027 (1)	0.6959 (2)
	NRWin300	0.6627 (3)	0.4784 (3)	0.3521 (3)	1.1537 (4)	0.8392 (5)	0.6294 (5)	1.4368 (5)	1.0398 (6)	0.7824 (6)
	NAWin1000	0.8730 (5)	0.6170 (5)	0.4411 (5)	1.1259 (3)	0.8300 (4)	0.6252 (4)	1.2807 (2)	0.9571 (3)	0.7341 (3)
	NAWin300	0.8812 (6)	0.6300 (6)	0.4537 (6)	1.1632 (5)	0.8638 (6)	0.6560 (6)	1.3382 (3)	1.0053 (4)	0.7757 (5)
	KWRin	1.4684 (7)	1.1272 (7)	0.8962 (7)	1.5578 (7)	1.1955 (7)	0.9503 (7)	1.6374 (6)	1.2568 (7)	0.9986 (7)

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A Survey on Concept Drift Adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014.

[3] V. Losing, B. Hammer, and H. Wersing, "KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift," in *ICDM*. IEEE Computer Society, 2016, pp. 291–300.

[4] ——, "Self-adjusting Memory: How to Deal with Diverse Drift Types," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, pp. 4899–4903.

[5] B. Pan, U. Demiryurek, and C. Shahabi, "Utilizing Real-World Transportation Data for Accurate Traffic Prediction," in *ICDM*. IEEE Computer Society, 2012, pp. 595–604.

[6] Y.-S. Jeong, Y.-J. Byon, M. M. Castro-Neto, and S. M. Easa, "Supervised Weighting-Online Learning Algorithm for Short-Term Traffic Flow Prediction," *Trans. Intell. Transport. Sys.*, vol. 14, no. 4, pp. 1700–1707, Dec. 2013.

[7] S. Van Vaerenbergh, J. Vía, and I. Santamaría, "Nonlinear System Identification using a New Sliding-Window Kernel RLS Algorithm," *Journal of Communications*, vol. 2, no. 3, pp. 1–8, May 2007.

[8] S. Yoshida, K. Hatano, E. Takimoto, and M. Takeda, "Adaptive Online Prediction Using Weighted Windows," *IEICE Transactions*, vol. 94-D, no. 10, pp. 1917–1923, 2011.

[9] P. Dell'Acqua, F. Bellotti, R. Berta, and A. D. Gloria, "Time-Aware Multivariate Nearest Neighbor Regression Methods for Traffic Flow Prediction," *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3393–3402, 2015.

[10] Y. Akihiro, M. Shigeru, I. Tatsuya, and U. Ken, "OPOSSAM: Online Prediction of Stream Data Using Self-adaptive Memory," in *IEEE International Conference on Big Data*, ser. Big Data '13. IEEE Computer Society, 2018, pp. 2365–2372.

[11] M. S. Dougherty and M. R. Cobbett, "Short-term inter-urban traffic forecasts using neural networks," *International Journal of Forecasting*, vol. 13, no. 1, pp. 21 – 31, 1997.

[12] T. Ban, R. Zhang, S. Pang, A. Sarrafzadeh, and D. Inoue, "Referential kNN Regression for Financial Time Series Forecasting," in *Neural Information Processing - 20th International Conference, ICONIP*, ser. Lecture Notes in Computer Science, vol. 8226. Springer, 2013, pp. 601–608.

[13] F. H. Al-Qahtani and S. F. Crone, "Multivariate k-nearest neighbour regression for time series data - A novel algorithm for forecasting UK electricity demand," in *The 2013 International Joint Conference on Neural Networks, IJCNN*, 2013, pp. 1–8.

[14] E. Ikonomovska, J. a. Gama, R. Sebastião, and D. Gjorgjevik, "Regression Trees from Data Streams with Drift Detection," in *Proceedings of the 12th International Conference on Discovery Science*, ser. DS '09. Springer-Verlag, 2009, pp. 121–135.

[15] E. Ikonomovska, J. a. Gama, and S. Džeroski, "Learning Model Trees from Evolving Data Streams," *Data Min. Knowl. Discov.*, vol. 23, no. 1, pp. 128–168, Jul. 2011.

[16] E. Ikonomovska and J. Gama, "Learning Model Trees from Data Streams," in *Proceedings of the 11th International Conference on Discovery Science*, ser. DS '08. Springer-Verlag, 2008, pp. 52–63.

[17] E. Ikonomovska, J. Gama, and S. Dzeroski, "Online tree-based ensembles and option trees for regression on evolving data streams," *Neurocomputing*, vol. 150, pp. 458–470, 2015.

[18] Y. Matsubara and Y. Sakurai, "Regime Shifts in Streams: Real-time Forecasting of Co-evolving Time Sequences," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. ACM, 2016, pp. 1045–1054.

[19] A. Bifet and R. Gavalda, "Learning from Time-Changing Data with Adaptive Windowing." in *SDM*, vol. 7. SIAM, 2007, pp. 443–448.

[20] H. Daumé, III, A. Kumar, and A. Saha, "Frustratingly Easy Semi-supervised Domain Adaptation," in *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, ser. DANLP 2010. Association for Computational Linguistics, 2010, pp. 53–59.

[21] F. Orabona, C. Castellini, B. Caputo, A. E. Fiorilla, and G. Sandini, "Model adaptation with least-squares SVM for adaptive hand prosthetics," in *2009 IEEE International Conference on Robotics and Automation, ICRA*, 2009, pp. 2897–2903.

[22] I. Kuzborskij and F. Orabona, "Stability and Hypothesis Transfer Learning," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. III–942–III–950.

[23] R. Gopalan, R. Li, V. M. Patel, and R. Chellappa, "Domain Adaptation for Visual Recognition," *Found. Trends. Comput. Graph. Vis.*, vol. 8, no. 4, pp. 285–378, Mar. 2015.

[24] G. Forman, "Tackling Concept Drift by Temporal Inductive Transfer," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '06. ACM, 2006, pp. 252–259.

[25] P. Zhang, B. J. Gao, X. Zhu, and L. Guo, "Enabling Fast Lazy Learning for Data Streams," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ser. ICDM '11. IEEE Computer Society, 2011, pp. 932–941.

[26] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional Concept Drift Detection and Density Synchronized Drift Adaptation," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, pp. 2280–2286.

[27] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala, "Locality-preserving Hashing in Multidimensional Spaces," in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '97. ACM, 1997, pp. 618–625.

[28] E. C. Neto, I. S. Jang, S. H. Friend, and A. A. Margolin, "The Stream Algorithm: Computationally Efficient Ridge-Regression via Bayesian Model Averaging, and Applications to Pharmacogenomic Prediction of Cancer Cell Line Sensitivity," in *Pacific Symposium on Biocomputing*, ser. Pac Symp Biocomput. World Scientific, 2014, pp. 27–38.

[29] J. Haworth, J. Shawe-Taylor, T. Cheng, and J. Wang, "Local online kernel ridge regression for forecasting of urban travel times," *Transportation Research Part C*, vol. 46, no. Complete, pp. 151–178, 2014.

[30] D. Dheeru and E. Karra Taniskidou, "UCI Machine Learning Repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml