Continuous Support Vector Regression for Nonstationary Streaming Data

Hang Yu, Jie Lu, *Fellow, IEEE*, and Guangquan Zhang, *Member, IEEE*

***Abstract*—Quadratic programming is the process of solving a special type of mathematical optimization problem. Recent advances in online solutions for quadratic programming problems (QPPs) have created opportunities to widen the scope of applications for support vector regression (SVR). In this vein, efforts to make SVR compatible with streaming data have been met with substantial success. However, streaming data with concept drift remain problematic because the trained prediction function in SVR tends to drift as the data distribution drifts. Aiming to contribute a solution to this aspect of SVR's advancement, we have developed continuous SVR (C-SVR) to solve regression problems with nonstationary streaming data, that is, data where the optimal input–output prediction function can drift over time. The basic idea of C-SVR is to continuously learn a series of input–output functions over a series of time windows to make predictions about different periods. However, strikingly, the learning process in different time windows is not independent. An additional similarity term in the QPP, which is solved incrementally, threads the various input–output functions together by conveying some learned knowledge through consecutive time windows. How much learned knowledge is transferred is determined by the extent of the concept drift. Experimental evaluations with both synthetic and real-world datasets indicate that C-SVR has better performance than most existing methods for nonstationary streaming data regression.**

***Index Terms*—Concept drift, continuous learning, streaming data, support vector regression (SVR).**

I. Introduction

THE AIM of a regression algorithm is to learn an input–output function $y = f(X)$ to make a prediction, where $X = \{x_1, x_2, \ldots, x_d\} \in R^d$ is the input, and y is the output [1]. As a parametric regression algorithm, support vector regression (SVR) can learn an $f(X)$ with good generalization performance [2], but learning $f(X)$ involves solving a single larger quadratic programming problem (QPP) [3]. If a dataset is designed as a persistent table, the solution can be based on the assumption that all the data are available and can be parsed multiple times [4], [5]. However, many datasets today take the form of streaming data—especially big datasets, such as in hand gesture recognition [6], wireless sensor networks [7], and social event detection [8]. In these applications, the datasets are typically an ordered sequence of data $\{\ldots, (X_{t-1}, y_{t-1}), (X_t, y_t), (X_{t+1}, y_{t+1}), \ldots\}$, where t represents the order of the datum's arrival, and $X_t \in R^d$ and $y_t \in R$. A datum (X_t, y_t) can only be read once [9]. In addition, the size of any data stream will far exceed the amount of space available for a single large QPP solution, so it is not possible for the solution to preserve all the data [10], [60].

These restrictions motivated many scholars to look for a new way to solve these QPPs. For example, Vijayakumar and Wu [11] proposed a sequential online greedy method for training SVR. This method incorporates a selection mechanism into a sequential SVR [12], [13] so that only a single datum is observed at each time step, and $f(X)$ is trained solely on that series of data. Then, the $f(X)$ was learned by incrementally adjusting the maximum margin distance between support vectors. This solution is fine for stationary streaming data. However, concept drift in streaming data is common in practice, that is, the data distribution $P_t(X, y) \neq P_{t+1}(X, y)$ will occur at timestamp $t + 1$, which leads to $f_t(X) \neq f_{t+1}(X)$. Normally, concept drift can be classified into four types [22], [23]: 1) sudden drift, which means a concept abruptly changes over a short period of time; 2) gradual drift, a new concept gradually replaces an old one over a period of time; 3) incremental drift, which means the old concept incrementally changes into a new concept over a long period of time; and 4) recurring drift, an old concept may reoccur after some time. If streaming data contain one or more concept drifts, it can be called nonstationary streaming data [32]–[34].

In this article, we propose continuous SVR (C-SVR) for nonstationary streaming data. Like an ensemble-based method, in C-SVR, a series of $f_i(X)$ is continuously learned in a series of time windows tw_i to determine the relationship between the input and output at different timestamps. However, only one $f_i(X)$ is saved in memory to make a prediction. A new $f_i(X)$ is learned in the new tw_i, and the old $f_{i-1}(X)$, which was learned in the last tw_{i-1}, is discarded. In addition, in contrast to algorithms [26], [27] that forget all learned knowledge, $f_i(X)$ learning is not independent in C-SVR. A similarity term added to the QPP carries some learned knowledge from $f_{i-1}(X)$ forward into $f_i(X)$. How much learned knowledge is transferred depends on the degree of the concept drift. Further, because

Manuscript received June 24, 2019; revised February 17, 2020 and June 19, 2020; accepted August 5, 2020. Date of publication September 10, 2020; date of current version May 19, 2022. This work was supported by the Australian Research Council under Discovery Grant DP190101733. This article was recommended by Associate Editor J. Jayadeva. *(Corresponding author: Hang Yu.)*

The authors are with the Decision Systems and e-Service Intelligence Lab, Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: hang.yu@student.uts.edu.au; jie.lu@uts.edu.au; guangquan.zhang@uts.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/10.1109/TCYB.2020.3015266.

Digital Object Identifier 10.1109/TCYB.2020.3015266

the data in nonstationary streaming data arrive sequentially, the QPP in C-SVR is solved incrementally.

Thus, the four main contributions of this article are given as follows.

1) A continuous learning strategy for SVR where $f(X)$ is learned from new data and some outdated learned knowledge is forgotten. Hence, $f(X)$ is not fixed; it varies in different time windows tw_i to adaptively suit the drift.
2) A new definition of a QPP that incorporates a similarity term to transfer learned knowledge from $f_{i-1}(X)$ into $f_i(X)$. Therefore, learning a new $f_i(X)$ is dependent on $f_{i-1}(X)$, which prevents catastrophic forgetting of learned knowledge of $f_{i-1}(X)$ [35].
3) An online way of solving the QPP of C-SVR based on the Karush–Kuhn–Tucker (KKT) conditions and one datum per time step, which means C-SVR can handle streaming data.
4) An automated method of deciding how much learned knowledge in $f_{i-1}(X)$ needs to be transferred to $f_i(X)$. As a result, C-SVR can forget different degrees of learned knowledge according to different types of concept drift.

The remainder of this article is organized as follows. Section II gives a brief overview of the forgetting mechanism. Section III describes the details of the C-SVR method, including the continuous learning strategy, the QPP of C-SVR, how to incrementally solve the QPP, and how to detect concept drift. The experiments are presented in Section IV. The concluding remarks and future works are given in Section V.

II. Related Work

Solving regression problems in nonstationary streaming data requires more than just considering the new information. A forgetting mechanism [24], [25] also needs to be designed to discard learned knowledge that is now outdated. To the best of our knowledge, the forgetting mechanisms largely fall into one of the two types.

A. Without Explicit Drift Detection

In this type of forgetting mechanism, the forgetting of learned knowledge is independent to the detection result of concept drift. In contrast, the forgetting strategy is triggered as long as new data arrive, thereby discarding outdated knowledge. For example, Wang *et al.* [19] proposed a budgeted stochastic gradient descent (BSGD) for training support vector machines. In BSGD, the weight w_i of model $f_i(X) = w_i X$ is updated when a new datum arrives, and a new support vector is added. If the number of support vector greater than the user-decided budget, one outdated support vector will be deleted and the weight w_i is readjusted. However, setting the value of the budget is a difficult task in itself. Ma *et al.* [14] proposed an accurate online SVR (AON-SVR) algorithm that follows the idea of [15]. In AON-SVR, three sets—the remaining, error, and supporting sets—are built according to the KKT conditions [16]. When a new datum arrives, it is assigned to one set through an infinite number of discrete steps until it meets the KKT conditions, while ensuring that the existing data continue to satisfy the KKT conditions at each step. Furthermore, a decremental algorithm is used to forget one datum from one of the three sets and then update $f(X)$. Following the idea of AON-SVR, Gu *et al.* [17] proposed an incremental v-SVR algorithm and extended the incremental v-SVR for ordinal regression [20]. In addition, to handle uncertain streaming data, Yu *et al.* [21] proposed an online robust SVR algorithm as an extension to the incremental v-SVR. Here, the adjustment speed of the model is too slow to keep up with sudden drift. Omitaomu *et al.* [27] proposed an incremental SVR with varying parameters (VPI-SVR) rather than fixed ones. VPI-SVR uses different parameters to learn $f(X)$ in a different time window tw_i. However, when $f(X)$ is stable in conjoint time windows, VPI-SVR uses the different parameters to learn $f(X)$, and the performance suffers as a result.

B. With Explicit Drift Detection

In the second type of forgetting mechanism, the learned knowledge is forgotten according to the detection result of concept drift. This type forms the largest category of forgetting mechanism and is used in many algorithms. For example, fast and incremental model trees (FIMT-DD), initially presented in [53], are the main example. Similar to standard Hoeffding Trees [54], features are ranked according to their variance in FIMT-DD, and if the two best-ranked features differ by at least the Hoeffding Bound [44], the tree branches and the process are repeated. FIMT-DD also encompasses a change detection scheme that periodically flags and adapts sub-branches of the tree where significant variance increases are observed. Adaptive model rules (AMRules) [55] is another relevant representative of streaming data regression. AMRules learns both ordered and unordered rule sets from data streams. To detect and adapt to concept drifts, each rule is associated with a Page-Hinkley drift detector [43], which prunes the rule set given changes in the incoming data. Liu and Zio [26] proposed an algorithm called feature vector selection to detect drift and a new SVR based on the new data is learned when concept drift is detected. However, it is not always easy to detect concept drift because its underlying causes are not necessarily evident in the data. In contrast to the above model-based methods, IBLStreams [48] is an instance-based learning algorithm, so the prediction result can be estimated by the weighted mean of the k neighbor instances. IBLStreams also uses a change detection scheme to update parameters, such as the k and kernel width.

Another important branch of this type of forgetting strategy is based on ensemble [29]–[31], that is, a new learner will be trained after concept drift is detected. For example, Zhai *et al.* [56] ensembled BSGD with a different budget, Ikonomovska *et al.* [57] ensembled online option trees for regression (ORTO), and Gomes *et al.* [58] proposed an adaptive random forest (ARF-Reg) by assembling FIMT-DD. The overall idea of ensemble-based methods is to learn a series of $f_i(X)$ directly from a series of time windows tw_i, so $f_i(X)$ essentially represents separate streaming data at different periods. However, as more and more $f_i(X)$ are saved, their size will eventually exceed the computer's memory. Therefore, these approaches include a method to select which $f_i(X)$ should be forgotten. However, designing a selection method is a difficult task in itself, especially since different types of concept drifts can co-exist in streaming data. Furthermore, ensemble-based methods normally rely on the same forgetting mechanism for different types of concept drift. Yet ignoring the differences between different types of concept drift will mean reduced performance. Moreover, each $f_i(X)$ for the current time window tw_{i} is constructed independently from the last time window $tw_{\mathrm{i}-1}$. However, as for nonstationary streaming data with gradual or incremental drift, as the old concept incrementally changes into a new concept over a long period of time, so $f_i(X)$ for the current time window tw_{i} can be similar to the last time window $tw_{\mathrm{i}-1}$ and they are not independent. This results in ensemble-based methods having drawbacks when handling nonstationary streaming data with gradual or incremental drift.

III. Our Method

This section demonstrates how our proposed C-SVR integrates new information from streaming data and the process for forgetting outdated information to adapt to concept drift. First, we provide a general overview of the continuous learning strategy, then each of the components is described in more detail in the individual sections.

A. Support Vector Regression

Given a batch dataset $D = \{(X_1, y_1), \ldots, (X_i, y_i), \ldots, (X_n, y_n)\}$ with n data, where $X_i \in R^d$ and $y_i \in R$, the classical ε-SVR [2] considers the following primal problem:

$$\begin{aligned} \min_{\omega,\varepsilon,b,\xi_i^{(*)}} \quad & \frac{1}{2}\|w\|^2 + C \cdot \left(v\varepsilon + \frac{1}{l}\sum_{i=1}^{l}\left(\xi_i + \xi_i^*\right) \right) \\ \text{s.t.} \quad & (\langle w, \phi(X_i)\rangle + b) - y_i \le \varepsilon + \xi_i \\ & y_i - (\langle w, \phi(X_i)\rangle + b) \le \varepsilon + \xi_i^* \\ & \xi_i^{(*)} \ge 0,\ \varepsilon \ge 0,\ i = 1, \ldots, l \end{aligned} \tag{1}$$

where the training data X_i are mapped into a high-dimensional reproducing kernel Hilbert space (RKHS) [36] using a transformation function Φ. w represents the weight of X_i, and $\|wt\|$ is a regularization term that characterizes the complexity of the regression model. b represents the bias and $\langle\cdot,\cdot\rangle$ denotes the inner product in the RKHS. C is a regularization constant, and $\xi^{(*)}$ is the slack variable. ("$*$" is shorthand for the variables both with and without asterisks.)

The corresponding dual is then

$$\begin{aligned} \min_{\alpha,\alpha^*} \quad & \frac{1}{2}\sum_{i,j=1}^{l}\left(\alpha_i^* - \alpha_i\right)\left(\alpha_j^* - \alpha_j\right)Q_{ij} + \varepsilon\sum_{i=1}^{l}\left(\alpha_i + \alpha_i^*\right) \\ & - \sum_{i=1}^{l}\left(\alpha_i^* - \alpha_i\right)Y_i \\ \text{s.t.} \quad & \sum_{i=1}^{l}\left(\alpha_i^* - \alpha_i\right) = 0 \\ & 0 \le \alpha_i^{(*)} \le C,\ i = 1, \ldots, l \end{aligned} \tag{2}$$

where $Q_{ij} = \langle\Phi(X_i), \Phi(X_j)\rangle = K(X_i, X_j)$. Here, $K(X_i, X_j)$ is a kernel function [37].

Once the corresponding dual is solved, the SVR will generate predictions using the following formula:

$$f(X) = \sum_{i=1}^{l}\theta_i K(X, X_i) + b \tag{3}$$

where $\theta_i = (\alpha_i^* - \alpha_i)$ presents the weight of the kernel.

B. Continuous Learning Strategy

The learning strategy of our proposed C-SVR is unique. It is not like the learning strategies of other online SVR-based methods such as [19], which use a constant state, that is, the same parameters, to learn the predictors, as the learning state of C-SVR is adaptive and dependent on the degree of concept drift. It also is not like the learning strategies of other ensemble SVR-based methods such as [56], which need to save multiple learners. In C-SVR, only one learner is saved.

Considering nonstationary streaming data of $\{(X_1, y_1), \ldots, (X_t, y_t), \ldots\}$ where t is the order the data arrives in, X_t is the input, and y_t is the output. Then, assume that a time window tw_{i} has size l so that a total of l data that can be captured in order by tw_{i}. For example, $tw_1 = \{(X_1, y_1), \ldots, (X_t, y_t), \ldots, (X_l, y_l)\}$, where $1 \le t \le l$. Our continuous learning strategy is based on this assumption and is shown in Fig. 1 (see page 4). From Fig. 1, we can see that a series of $f_i(X)$ is learned in consecutive time windows tw_{i} to make predictions about the near future. Note that nonstationary streaming data in consecutive tw_{i} do not overlap. C-SVR's learning process in each tw_{i} is further divided into the following steps.

Step 1: The data distribution of $tw_{\mathrm{i}-2}$ is compared to $tw_{\mathrm{i}-1}$ to determine the degree of concept drift, then the learned knowledge in $f_{i-1}(X)$ is transferred to $f_i(X)$.

Step 2: One new datum X_t is taken.

Step 3: The incremental solution is used to update $f_i(X_{t-1})$ to $f_i(X_t)$ based on: the new datum X_t; the degree of concept drift; and the learned knowledge of $f_{i-1}(X)$.

Step 4: If a new time window $tw_{\mathrm{i}+1}$ does not arrive, go to step 2. Otherwise, go to step 1.

Each time a new tw_{i} arrives, this learning process repeats iteratively to learn the current $f_i(X)$ and continuously adapt to the current concept. However, in the first- and second-time windows, tw_1 and tw_2, the learning process is slightly different. In the first tw_1, we do not need to consider the learned knowledge because we do not have any learned knowledge. In addition, in the first tw_1 and second tw_2, we do not need to consider the degree of concept drift, the reason being that the degree of concept drift is obtained by comparing the $f_i(X)$ and $f_{i+1}(X)$ in our proposed C-SVR, so it is only when we have learned at least two functions that we can determine the degree of drift.

C. Quadratic Programming Problem

Based on the above section, we know that C-SVR should be able to continuously learn a series of $f_i(X)$ for consecutive

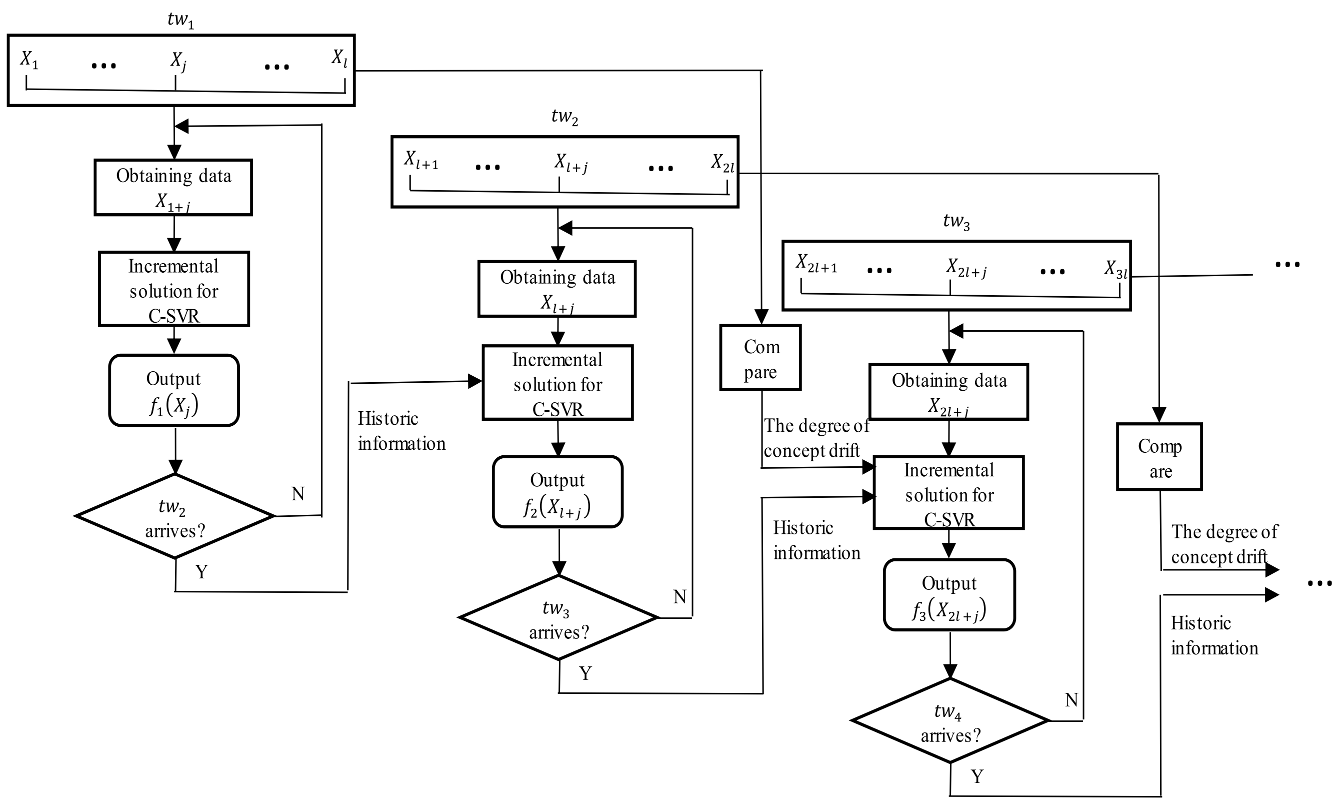
tw_1
X_1 ••• X_j ••• X_l
Obtaining data X_{1+j}
Incremental solution for C-SVR
Output $f_1(X_j)$
tw_2 arrives?
N
Y
Historic information
tw_2
X_{l+1} ••• X_{l+j} ••• X_{2l}
Obtaining data X_{l+j}
Incremental solution for C-SVR
Output $f_2(X_{l+j})$
tw_3 arrives?
N
Y
Com pare
The degree of concept drift
Historic information
tw_3
X_{2l+1} ••• X_{2l+j} ••• X_{3l}
Obtaining data X_{2l+j}
Incremental solution for C-SVR
Output $f_3(X_{2l+j})$
tw_4 arrives?
N
Y
Comp are
The degree of concept drift
Historic information
•••

Fig. 1. Continuous learning strategy.

time windows tw_i, and also that learning $f_i(X)$ is dependent on $f_{i-1}(X)$ and the degree of concept drift. Thus, when the data distribution of tw_{i-1} drifts slightly relative to the data distribution of tw_{i-2}, $f_i(X)$ should be more similar to $f_{i-1}(X)$ and less similar otherwise. Therefore, the best solution for our expectation is a compromise between (individual) optimality and (neighbor) similarity. Accordingly, we have defined a simple distance in C-SVR to measure $d(f_{i-1}(X), f_i(X))$ so as to quantify the similarity between two neighboring input–output functions $f_{i-1}(X)$ and $f_i(X)$ and, then, to minimize the two-term cost function

$$\min Err_i^2 + \gamma \cdot d(f_i, f_{i-1}) \tag{4}$$

where the first term is the usual cost function for C-SVR and the second evaluates the total difference between the two consecutive discriminant functions. Here, the free parameter γ regulates the compromise between both terms as with any regularized fitting.

Similar to the classical ε-SVR method [2], we look for a pair of (w_i, b_i) that define an input–output function given by $f_i(X) = \langle w_i \cdot \Phi(X)\rangle + b_i$, where b_i can be computed as follows:

$$b_i = y_i - \langle w_i \cdot \Phi(X)\rangle. \tag{5}$$

Hence, a simple quadratic distance measure is used to quantify the diversity between functions

$$d(f_i, f_{i-1}) = \|w_i - w_{i-1}\|^2. \tag{6}$$

Applying this measure to (4) results in a new QPP for C-SVR, introduced as follows:

$$\begin{aligned}\min_{\omega_i,\varepsilon,b_i,\xi(*)}\ & \frac{1}{2}\|w_i\|^2 + C\cdot\sum_{t=1}^{l}\left(\xi_{it}+\xi_{it}^*\right) + \frac{\gamma}{2}\left(\|w_i - w_{i-1}\|^2\right)\\ \text{s.t.}\ & -y_{it} + (\langle w_i, \Phi(X_{it})\rangle + b_i) + \varepsilon_i + \xi_{it} \ge 0\\ & y_{it} - (\langle w_i, \Phi(X_{it})\rangle + b_i) + \varepsilon_i + \xi_{it}^* \ge 0\\ & \xi_{it}^{(*)} \ge 0,\ \varepsilon_i \ge 0,\ t = 1,\dots,l.\end{aligned} \tag{7}$$

The first two terms in (7) correspond to the usual margin and absolute penalty terms in an SVR. The final term in (7) corresponds to the new diversity penalty. Including this term means $f_i(X)$ is dependent on $f_{i-1}(X)$. Hence, the solution to this optimization problem with dual variables is to find the saddle point of the Lagrangian [38]

$$\begin{aligned}L_D =\ & \frac{1}{2}\|w_i\|^2 + C\cdot\sum_{t=1}^{l}\left(\xi_{it}+\xi_{it}^*\right) + \frac{\gamma}{2}\left(\|w_i - w_{i-1}\|^2\right)\\ & - \sum_{t=1}^{l}\alpha_{it}(y_{it} - \langle w_i, \Phi(X_{it})\rangle - b_i + \varepsilon_i + \xi_{it})\\ & - \sum_{t=1}^{l}\alpha_{it}^*\left(\langle w_i, \Phi(X_{it})\rangle + b_i - y_{it} + \varepsilon_i + \xi_{it}^*\right)\\ & - \sum_{t=1}^{l}\left(\eta_{it}\xi_{it} + \eta_{it}^*\xi_{it}^*\right)\end{aligned} \tag{8}$$

where $\alpha_{it}^{(*)}$ and $\eta_{it}^{(*)}$ are non-negative Lagrange multipliers. Differentiating L with respect to w_i, b_i, and $\xi_{it}^{(*)}$ and setting the result to zero yields

$$\begin{aligned}\frac{\partial L_D}{\partial w_i} &= w_i + \gamma(w_i - w_{i-1}) - \sum_{t=1}^{l}\Phi(X_{it})\left(\alpha_{it} - \alpha_{it}^*\right) = 0\\ &\Rightarrow w_i = \frac{\gamma}{1+\gamma}w_{i-1} + \frac{1}{1+\gamma}\sum_{t=1}^{l}\Phi(X_{it})\left(\alpha_{it} - \alpha_{it}^*\right)\\ \frac{\partial L_D}{\partial b_i} &= -\sum_{t=1}^{l}\left(\alpha_{it} - \alpha_{it}^*\right) = 0\\ \frac{\partial L_D}{\partial \zeta_{it}} &= C - \eta_{it} - \alpha_{it} = 0 \Rightarrow \alpha_{it} \in [0, C]\\ \frac{\partial L_D}{\partial \zeta_{it}^*} &= C - \eta_{it}^* - \alpha_{it}^* = 0 \Rightarrow \alpha_{it}^* \in [0, C].\end{aligned} \tag{9}$$

Substituting (9) into L leads to the following dual problem:

$$\begin{aligned}\min_{\alpha^{(*)}}\ & \frac{1}{2(1+\gamma)}\sum_{t=1}^{l}\sum_{j=1}^{l}Q_{tj}\left(\alpha_{it} - \alpha_{it}^*\right)\left(\alpha_{ij} - \alpha_{ij}^*\right)\\ & + \frac{\gamma}{1+\gamma}\sum_{t=1}^{l}\left(\alpha_{it} - \alpha_{it}^*\right)y_{(i-1)t}\\ & - \sum_{t=1}^{l}y_{it}\left(\alpha_{it} - \alpha_{it}^*\right) + \sum_{t=1}^{l}\varepsilon_i\left(\alpha_{it} + \alpha_{it}^*\right)\\ \text{s.t.}\ & \alpha_{it}, \alpha_{it}^* \in [0, C],\ t = 1, \ldots, l.\\ & \sum_{t=1}^{l}\left(\alpha_{it} - \alpha_{it}^*\right) = 0.\end{aligned} \tag{10}$$

Here, Q_{ij} is a matrix with kernel properties

$$Q_{tj} = \left\langle \Phi(X_{it}) \cdot \Phi\left(X_{ij}\right)\right\rangle = K\left(X_{it}, X_{ij}\right) \tag{11}$$

where K is a kernel function.

D. Incremental Solution

With nonstationary streaming data, one typically operates with only one datum at a time. Therefore, we propose an incremental solution to solve the QPP of C-SVR on a data-by-data basis. As shown in the previous section, minimizing α, α^* values create a dual optimization problem (8) that must be solved. Similarly, we can compute another Lagrange function that includes all constraints and introduces other Lagrange multiples

$$\begin{aligned}L_D = & \frac{1}{2(1+\gamma)}\sum_{t=1}^{l}\sum_{j=1}^{l}Q_{tj}\left(\alpha_{it} - \alpha_{it}^*\right)\left(\alpha_{ij} - \alpha_{ij}^*\right)\\ & + \frac{\gamma}{1+\gamma}\sum_{t=1}^{l}\left(\alpha_{it} - \alpha_{it}^*\right)y_{(i-1)t}\\ & - \sum_{t=1}^{l}y_{it}\left(\alpha_{it} - \alpha_{it}^*\right) + \sum_{t=1}^{l}\varepsilon_{it}\left(\alpha_{it} + \alpha_{it}^*\right)\\ & + \sum_{t=1}^{l}\left[\mu_{it}(\alpha_{it} - C) + \mu_{it}^*\left(\alpha_{it}^* - C\right)\right]\\ & + \zeta_i\sum_{t=1}^{l}\left(\alpha_{it} - \alpha_{it}^*\right) - \sum_{t=1}^{l}\left(\delta_{it}\alpha_{it} + \delta_{it}^*\alpha_{it}^*\right)\\ \text{s.t.}\quad & \delta_{it}^{(*)}, \mu_{it}^{(*)}, \zeta_i \geq 0, t = 1, \ldots, l.\end{aligned} \tag{12}$$

Computing partial derivatives of the Lagrangian

$$\begin{aligned}\frac{\partial L_D}{\partial \alpha_{it}} &= \frac{1}{2(1+\gamma)}\sum_{j=1}^{l}Q_{tj}\left(\alpha_{ij} - \alpha_{ij}^*\right) + \frac{\gamma}{1+\gamma}y_{(i-1)t}\\ &\quad - y_{it} + \varepsilon_i - \delta_{it} + \mu_{it} + \zeta_i = 0\\ \frac{\partial L_D}{\partial \alpha_{it}} &= -\frac{1}{2(1+\gamma)}\sum_{j=1}^{l}Q_{tj}\left(\alpha_{ij} - \alpha_{ij}^*\right) - \frac{\gamma}{1+\gamma}y_{(i-1)t}\\ &\quad + y_{it} + \varepsilon_i - \delta_{it}^* + \mu_{it}^* - \zeta_i = 0\\ \frac{\partial L_D}{\partial \zeta_i} &= \sum_{j=1}^{l}\left(\alpha_{ij} - \alpha_{ij}^*\right) = 0.\end{aligned} \tag{13}$$

Then to increase readability, we can replace ξ_i with $b : \xi_i = b$. This is still an optimization problem with a convex domain, so the sufficient conditions for a point to be optimum are KKT [16]

$$\begin{aligned}& \alpha_{it} \in [0, C]\\ & \alpha_{it}\left(\sum_{j=1}^{l}Q'_{tj}\left(\alpha_{ij} - \alpha_{ij}^*\right) + b_i + y'_{(i-1)t} + \varepsilon_i - y_{it} - \delta_{it} + u_{it}\right)\\ &\quad = 0\\ & \alpha_{it}^* \in [0, C]\\ & \alpha_{it}^*\left(-\sum_{j=1}^{l}Q'_{tj}\left(\alpha_{ij} - \alpha_{ij}^*\right) - b_i - y'_{(i-1)t} + \varepsilon_i + y_{it} - \delta_{it}^* + u_{it}^*\right)\\ &\quad = 0\\ & \delta_{it} \geq 0, \quad \delta_{it}\alpha_{it} = 0\\ & \delta_{it}^* \geq 0, \quad \delta_{it}^*\alpha_{it}^* = 0\\ & \mu_{it} \geq 0, \quad \mu_{it}(\alpha_{it} - C) = 0\\ & \mu_{it}^* \geq 0, \quad \mu_{it}^*\left(\alpha_{it}^* - C\right) = 0\end{aligned} \tag{14}$$

where

$$Q'_{tj} = \frac{1}{2(1+\gamma)}Q_{tj} \tag{15}$$

and

$$y'_{(i-1)t} = \frac{\gamma}{(1+\gamma)}y_{(i-1)t} \tag{16}$$

Further, if we assume the estimated $f_i(X_t)$ in the tw_i can be written as $f'_i(X_t) = \sum_{j=1}^{l}Q'_{tj}(\alpha_{ij} - \alpha_{ij}^*) + b_i$, then a marginal function can be obtained

$$h_i(X_t) = \left(f'_i(X_t) + y'_{(i-1)t}\right) - y_{it}. \tag{17}$$

For simplicity, θ_{ij} can be defined as the difference between α_{ij} and α_{ij}^*, and the KKT conditions can be obtained by replacing θ_{ij} in (13)

$$\begin{cases} h_i(X_t) \geq +\varepsilon, & \theta_{ij} = -C\\ h_i(X_t) = +\varepsilon, & \theta_{ij} \in [-C, 0]\\ h_i(X_t) \in [-\varepsilon, +\varepsilon], & \theta_{ij} = 0\\ h_i(X_t) = -\varepsilon, & \theta_{ij} \in [0, C]\\ h_i(X_t) \leq -\varepsilon, & \theta_{ij} = +C.\end{cases} \tag{18}$$

Using these KKT conditions, the data can be divided into three sets.

Algorithm 1 Incremental Solution

```
Input: a new data $(X_c, y_c)$
Parameter: initial $\theta_{ic} = 0$, $b_i = 0$
Output: $f_i(X)$
1:  Compute $h_i(X_c)$
2:  if $|h_i(X_c)| < \varepsilon_i$:
3:     Add $X_c$ to the remaining set
4:     Exit
4:  else:
5:   while $X_c$ is not added into a set:
6:        Compute $\beta$ and $\omega$ according to (19) and (23)
7:        Compute the minimal increment $\Delta\theta_i$.
8:        Update $\Delta\theta_i$, $\Delta b_i$, $\Delta h_i(X_c)$, $S_S$, $E_S$ and $R_S$.
9:        Update the inverse matrix $R$.
10:   end while
11: end if
12: output $f_i'(X) + y'_{(i-1)t}$
```

1) *Support Set:*

$$S_S = \{t|(\theta_{ij} \in [0, +C] \wedge h_i(X_t) = -\varepsilon_i) \vee (\theta_{ij} \in [-C, 0] \wedge h_i(X_t) = +\varepsilon_i)\}.$$

2) *Error Set:*

$$E_S = \{t|(\theta_{ij} = -C \wedge h_i(X_t) \geq +\varepsilon_i) \vee (\theta_{ij} = +C \wedge h_i(X_t) \leq -\varepsilon_i)\}.$$

3) *Remaining Set:*

$$R_S = \{t|\theta_{it} = 0 \wedge |h_i(X_t)| \leq \varepsilon_i\}.$$

Hence, our proposed incremental solution aims to find a way to maintain consistent KKT conditions (15) when new data are added to one of the three sets.

The support set is defined as $S_S = \{s_1, s_2, \ldots, s_{l_s}\}$ and, to ensure all data satisfy the KKT conditions (15) during the learning procedure, the following linear system occurs when S_S is changed:

$$\sum_{j\in S_S} Q'_{tj}\Delta\theta_{ij} + \Delta b_i = -Q'_{tc}\Delta\theta_{ic}$$
$$\sum_{j\in S_S} \Delta\theta_{ij} = -\theta_{ic}. \quad (19)$$

These equations can be rewritten in an equivalent matrix form

$$\begin{bmatrix} \Delta b_i \\ \Delta\theta_{s_1} \\ \vdots \\ \Delta\theta_{s_{l_s}} \end{bmatrix} = \beta\Delta\theta_{ic} = \begin{bmatrix} \beta_b \\ \beta_{s_1} \\ \vdots \\ \beta_{s_{l_s}} \end{bmatrix}\Delta\theta_{ic} = -R\begin{bmatrix} 1 \\ Q'_{s_1c} \\ \vdots \\ Q'_{s_{l_s}c} \end{bmatrix}\Delta\theta_{ic} \quad (20)$$

where matrix R can be defined as

$$R = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q'_{s_1s_1} & \cdots & Q'_{s_1s_{l_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q'_{s_{l_s}s_1} & \cdots & Q'_{s_{l_s}s_{l_s}} \end{bmatrix}^{-1}. \quad (21)$$

For the error and remaining sets, we have defined a "not support set" as $N_S = E_S \cup R_S = \{n_1, n_2, \ldots, n_{l_n}\}$. Again, to ensure all data satisfy the KKT conditions (9) during the learning procedure, the following linear system occurs when N_S is changed:

$$\Delta h_i(X_t) = \sum_{j=1}^{l} Q'_{tj}\Delta\theta_{ij} + Q'_{tc}\Delta\theta_{ic} + \Delta b_i. \quad (22)$$

Rewriting the variation of the h formula in matrix notation gives

$$\begin{bmatrix} \Delta h_i(X_{n_1}) \\ \vdots \\ \Delta h_i(X_{n_{\ln}}) \end{bmatrix} = \omega\Delta\theta_{tc} \quad (23)$$

where matrix ω can be defined as

$$\omega = \begin{bmatrix} \Delta Q'_{n_1c} \\ \vdots \\ \Delta Q'_{n_{\ln}c} \end{bmatrix} + \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q'_{n_1s_1} & \cdots & Q'_{n_1s_{ls}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q'_{n_{\ln}s_1} & \cdots & Q'_{n_{\ln}s_{ls}} \end{bmatrix}\beta. \quad (24)$$

From (19) and (22), we can see that the values of $\Delta\theta_i$ and Δb_i can be updated by computing β, and the value of Δh_i can be updated by computing ω. Therefore, $f_i(X)$ can be learned with Algorithm 1.

Algorithm 1 shows that the minimal increment $\Delta\theta_i$ can be obtained by the same method used in [15]. The inverse matrix R is updated in two situations: 1) adding the first data or 2) adding other data.

When adding the first data in R, the update follows:

$$R = \begin{bmatrix} -Q'_{11} & 1 \\ 1 & 0 \end{bmatrix}. \quad (25)$$

When adding other data, R is updated with

$$R_{\text{new}} = \begin{bmatrix} & & & 0 \\ & R & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\omega}\begin{bmatrix} \beta \\ 1 \end{bmatrix}\begin{bmatrix} \beta^T & 1 \end{bmatrix} \quad (26)$$

where β and ω are recomputed to follow each situation. If one datum is added to the S_S, ω is defined as

$$\omega_i = Q'_{cc} + \begin{bmatrix} 1 & Q'_{cS_1} & \cdots & Q'_{cS_{ls}} \end{bmatrix}\beta. \quad (27)$$

As outlined, this incremental solution means C-SVR can handle streaming data because $f(X)$ is updated data-by-data.

E. Concept Drift

From (7), we know that the free parameter γ regulates the diversity of $f_i(X)$ and $f_{i-1}(X)$. Low values of γ will almost decouple $f_i(X)$ and $f_{i-1}(X)$, which allows for increased flexibility. Conversely, high γ values will produce an $f_i(X)$ that is almost identical to $f_{i-1}(X)$. Therefore, parameter γ can be seen as the degree of concept drift. Parameter γ should increase to increase the similarity of $f_i(X)$ to $f_{i-1}(X)$ because $f_{i-1}(X)$ should gradually approximate the real input–output function, which does not drift. In contrast, parameter γ should decrease to differentiate $f_i(X)$ from $f_{i-1}(X)$ because the real input–output function has drifted and $f_{i-1}(X)$ no longer represents it.

To determine the value of parameter γ, we have incorporated our previous work, that is, a competence-based drift detection method [41], into C-SVR. The reason for choosing this method is when using an SVR to learn $f(X)$, only the support vectors are saved because the $f(X)$ is constructed only by support vectors. Hence, only support vectors can be analyzed to identify concept drift. However, the number of support vectors is small. Our proposed competence-based drift detection method is able to achieve stable and good results, especially, for small samples [41]. Therefore, incorporating our proposed competence-based drift detection method results in our proposed C-SVR achieving more stable and better results than other concept drift detection methods [28], [39], [40].

The key idea of the competence-based drift detection method is that changes in the probability distribution of data should reflect its competence [42]. Hence, our proposed competence-based drift detection method compares two data distributions in the competence space instead of the original data feature space. Assume two datasets $S_1, S_2 \subseteq CB$(case-base), the competence-based empirical distance between S_1 and S_2 is defined as [41]

$$d^{CB}(S_1, S_2) = 2 \times \sup_{A \in \mathcal{A}} \left| S_1^{CB}(A) - S_2^{CB}(A) \right| \quad (28)$$

where $0 \leq d^{CB}(S_1, S_2) \leq 1$. Here, the more different the distribution of the two groups of data, the larger the competence-based empirical distance.

Therefore, to get the value of parameter γ, a group of support vectors S_{i-2} in tw_{i-2} and a group of support vectors S_{i-1} in tw_{i-1} will be joined together to form one CB. After modeling all the support vectors in CB in the competence model, two related closures $RC^{CB}(S_1)$ and $RC^{CB}(S_2)$ are separately obtained. Using the density of $RC_i^{CB}(S) \in RC^{CB}(S)$, each group of support vectors can be represented by their distribution in the competence space. The difference between the support vectors of the two groups can be measured by the competence-based empirical distance $d^{CB}(S_{i-1}, S_{i-2})$. However, the distance $d^{CB}(S_{i-1}, S_{i-2})$ cannot directly be used to set the γ_i because of the range of γ_i is $[0, +\infty]$, and if the degree of concept drift increases, the value of γ_i should decrease. Therefore, we use the following formulation to calculate γ_i:

$$\gamma_i = \begin{cases} +\infty, & \text{where } G\left(d^{CB}\right) = 0 \\ \frac{1}{G(d^{CB}(S_{i-1}, S_{i-2}))}, & \text{where } 0 < G\left(d^{CB}\right) < +\infty \\ 0, & \text{where } G\left(d^{CB}\right) = +\infty \end{cases} \quad (29)$$

where $G(d^{CB}(S_{i-1}, S_{i-2}))$ can be calculated by

$$G = \begin{cases} 0, & \text{where } d^{CB} = 1/2 \\ \frac{1}{1-1/(n-1)} - 1, & \text{where } d^{CB} = 1 - 1/n,\ n > 2 \\ 1/d^{CB}, & \text{otherwise.} \end{cases} \quad (30)$$

Although $d^{CB}(S_{i-1}, S_{i-2})$ is a time-consuming task, the calculation of $d^{CB}(S_{i-1}, S_{i-2})$ is fast in C-SVR because the number of support vectors in each group is small.

IV. Experiments

This section details the experimental results to compare C-SVR to the state-of-the-art approaches. The evaluations involve different scenarios using both artificial datasets and real-world datasets (see Table I). The artificial datasets allowed us to control the relevant parameters and to empirically evaluate the algorithms with specific types of concept drift. The real-world datasets enabled us to evaluate the merit of the proposed approach in practical scenarios. However, note that in these latter experiments, it was not always possible to precisely state when a drift occurred or even whether there was drift at a specific timestamp. All experiments were conducted in Python 3.5 on a PC running Windows 7 with an Intel Core i5 processor (2.40 GHz) and 8-GB RAM.

A. Experimental Setup

To validate the effectiveness of C-SVR, we compared its performance to AON-SVR [14], VPI-SVR [27], incremental SVR based on time windows (TW-SVR), AddExp.C-SVR [45], and Learn++.NSE-SVR [46].

AON-SVR is a popular online SVR algorithm for handling streaming data because its incremental solution of the QPP was widely used in other online SVR algorithms, including those mentioned above. For example, VPI-SVR and AON-SVR use the same incremental solution. The only difference between the two is that VPI-SVR uses different parameters in different time windows to learn $f_i(X)$. The incremental solution for TW-SVR is also the same as AON-SVR, but when a new time window arrives, the information of the previous $f_{i-1}(X)$ is completely discarded and a new $f_i(X)$ is learned. AddExp.C and Learn++.NSE are two well-known adaptive ensembles for dealing with concept drift. They are adaptive datum-based ensembles (AddExp.C) and adaptive batch-based ensembles (Learn++.NSE), respectively. In our experiments, we made a small change to these two ensemble algorithms. In AddExp.C-SVR, new ε-SVR were learned in the new tw_i through the incremental solution from AON-SVR. Then, the AddExp.C algorithm was used to update the weights of each expert prediction $f_i(X)$ to ultimately make the predictions. In contrast, each time step of the Learn++.NSE consists of a batch of data (these can be seen as the size of the time windows). Hence, the batches are considered to have a size m. The weak learner is the ε-SVR. However, Learn++.NSE-SVR does not learn a new ensemble $f_i(X)$ in a new tw_i. Rather the Learn++.NSE algorithm is used to decide whether to learn a new ensemble, but the $f_i(X)$ is also updated in a tw_i by the incremental solution, which is used in AON-SVR for handling streaming data. Learn++.NSE is also used to update the weights of each ensemble $f_i(X)$ to make a prediction.

All the above-mentioned online SVR algorithms are designed with a forgetting strategy. AON-SVR includes a decremental learning algorithm [15]. When the number of processed data exceeds a certain number, the information of one previous data is ignored to update $f_i(X)$. The "certain number" can be seen as the size m of a time window. VPI-SVR and TW-SVR use a simple forgetting strategy, that is, the previous $f_{i-1}(X)$ is completely forgotten and a new $f_i(X)$ is learned when a new tw_i arrives. With the ensemble SVR algorithms, the forgetting strategy used follows the best result indicated by its authors. With AddExp.C-SVR, the $f_i(X)$ with the lowest weight was excluded. With Learn++.NSE-SVR, the model j

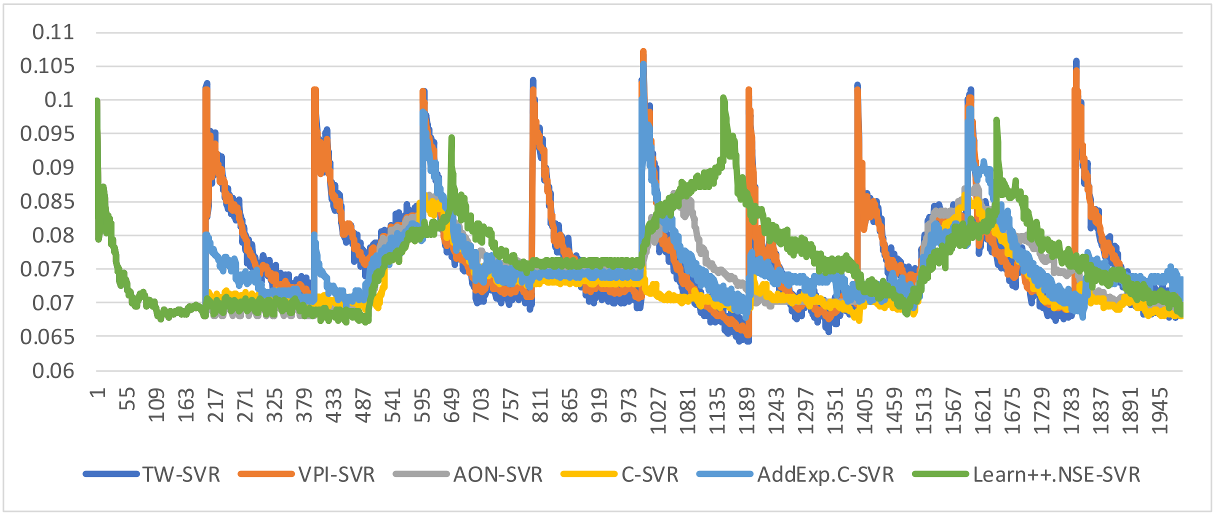

Fig. 2. Results on the drifting hyperplane dataset (the x-axis represents the size of instances, and the y-axis represents RMSE).

with the largest current error was excluded. In all ensemble-based SVR algorithms, the maximum number of models in the ensemble was set to 20. This choice was considered the most suitable for all the ensembles since the maximum number of models usually varies between 15 and 30 [45]–[47], and the use of more models linearly increases the processing time of the experiments.

Turning to the parameters of the above algorithms, two parameters C and ε needed to be set in all algorithms, each of which has been shown to affect performance [18]. However, the automatic method [27] of setting parameters C and ε in VPI-SVR means that only two predefined parameters C and ε needed to be set for the first-time window tw_1. In subsequent time windows, these parameters were set automatically. Some of the other parameters in the ensemble SVRs needed to be set for ensemble learning. For example, in AddExp.C, the parameters were set based on the pilot studies from [45]: factor for decreasing weights $\beta = 0.5$, factor for new expert weight $\gamma = 0.1$, and loss required to add a new expert $\tau = 0.5$. The parameters for Learn++.NSE were set according to the authors' suggestions [48]: slope $a = 0.5$ and infliction point $b = 10$.

B. Artificial Datasets

To illustrate that our method C-SVR has better performance to deal with nonstationary streaming data, this section presents some experimental results for the artificial datasets with concept drift. However, in this article, we only consider datasets with incremental, sudden, and gradual drift in the following experiments (see Table I), the reason being that research into these three types of drift detection focuses on how to detect the concept transformation process rapidly. In contrast, the study of recurring drift emphasizes the use of historical concepts—that is, how to find the best matched historical concepts in the shortest time.

The first is the drifting hyperplane dataset [49]. This is a well-known drift dataset for evaluating algorithms that deal with concept drift [27], [29]. It contains noise and incremental drifts and nonrecurring drifts and is similar to the one proposed in [45] (AddExp). The entire dataset consists of ten inputs with a uniform distribution over the interval of [0, 1] and one output data $y_i \in [0, 1]$; and there are 2000 data ($M = 2000$) in the dataset. The dataset contains four concepts, with each concept holding 500 data. A random variate noise uniformly distributed in the interval of $[-0.1, 0.1]$ is injected into each output y_i (for $i = 1, \ldots, M$). The value of y_i is set to 0 or 1 if its value is less than 0 or greater than 1, respectively.

In this experiment, the size of the time window is 200. Then, we set $C = 100$ and $\varepsilon = 0.1$ for each of the above-mentioned algorithms. Each method is evaluated using the root-mean-square error (RMSE), which is calculated using the predicted outputs and the real outputs from the online data. RMSE is a widely used metric to evaluate models. It provides a loss function that penalizes larger errors. A lower RMSE means the algorithm performed better. Fig. 2 shows the results of our comparison based on the drifting hyperplane dataset.

From Fig. 2, in general, we can see that, in most time windows that have no incremental drift, such as tw_1 ([0-200]), tw_2 ([200-400]), tw_7 ([1200-1400]), and tw_{10} ([1800-200]), the RMSE obtained by each online SVR-based algorithm shows a convergent trend as the training data increases. However, when incremental drift occurs, that is, $M = 500$ and $M = 1500$, the RMSE obtained by every online SVR-based algorithm shows an upward tendency as the training data increases. The trend of the RMSE is identical because each online SVR-based algorithm learns $f_1(x)$ with the same parameters in the tw_1 ([0-200]). However, because TW-SVR and VPI-SVR relearn a new function $f_i(x)$ in each new tw_i, such as tw_2 ([200-400]), the RMSE fluctuates sharply when each tw_i arrives, that is, RMSE instantly surges and then gradually declines to final convergence. Although AddExp.C-SVR also relearns a new function $f_i(x)$ in each new tw_i, the old function $f_{i-1}(x)$ is not discarded, so the RMSE does not sharply increase. In contrast, AON-SVR and Learn+.NSE-SVR do not relearn a new function $f_i(x)$ when a new tw_i arrives, so the

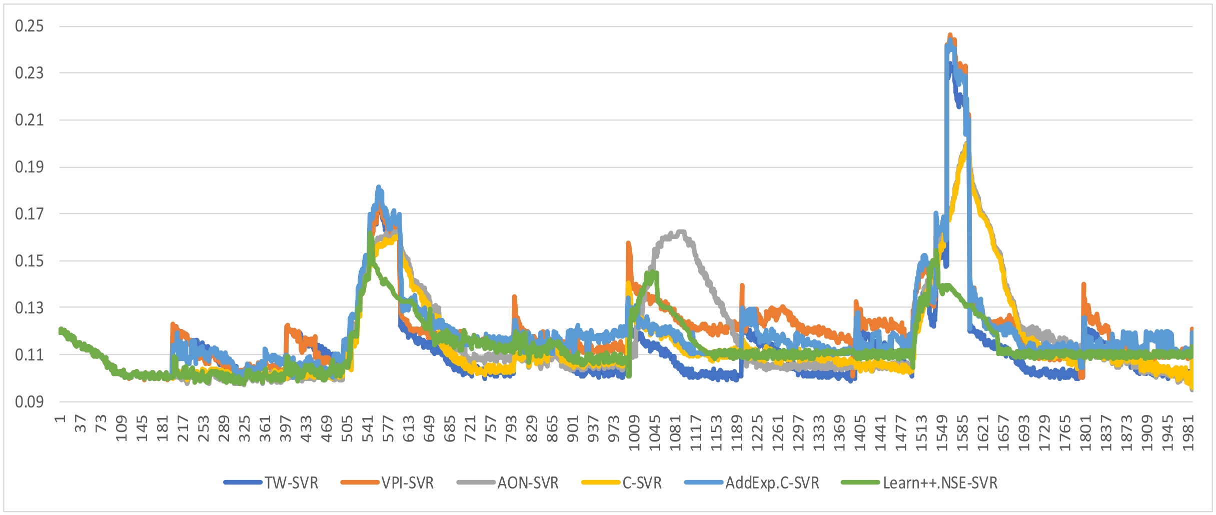

Fig. 3. Results on the drifting Friedman's dataset (the x-axis represents the size of instances, and the y-axis represents RMSE).

RMSE remains stable. However, when the first time window tw_i arrives after an incremental drift, such as tw_4 ([600-800]) and tw_9 ([1600-1800]), the RMSE convergence speed for AON-SVR and Learn++.NSE-SVR is slower than that of TW-SVR, and VPI-SVR, and AddExp.C-SVR. However, each has a different reason for converging more slowly. For AON-SVR, the reason is that the information of the last $f_{i-1}(x)$ is not completely discarded. However, for Learn++.NSE-SVR, the reason is that the incremental drift cannot be detected in time, so the RMSE will continue to grow until Learn+.NSE-SVR decides to learn a new function. Also, the RMSE instantaneously surges at the moment of learning a new function. In addition, at time $M = 1000$, although incremental drift occurs, TW-SVR, VPI-SVR, and AddExp.C-SVR learn two distinct functions in tw_5 ([800-1000]) and tw_6 ([1000-1200]), and the RMSE obtained by TW-SVR, VPI-SVR, and AddExp.C-SVR is not influenced by concept drift, but the RMSE obtained by AON-SVR and Learn+.NSE-SVR is influenced by incremental drift, so it increases at that time. C-SVR also relearns a new function $f_i(x)$ in the new tw_i but, unlike the above algorithms, the information of the last function $f_{i-1}(x)$ is not completely discarded so the RMSE does not instantaneously surge when a new tw_i arrives. The RMSE only slightly trends up, such as in the early stages of tw_2 ([200-400]). However, when the first time window arrives after an incremental drift occurs, such as tw_4 ([600-800]) and tw_9 ([1600-1800]), C-SVR's RMSE convergence speed is slower than TW-SVR and VPI-SVR, but faster than AON-SVR, AddExp.C-SVR, and Learn+.NSE-SVR. In addition, although incremental drift occurs at time $M = 1000$, C-SVR's RMSE is not affected by the drift because it learns two distinct functions in tw_5 ([800-1000]) and tw_6 ([1000-1200]).

To further illustrate C-SVR's advantage, we experimented with a drifting Friedman's function as the sudden drift dataset. Friedman's function has linear and nonlinear relations between the input variables and the output variable. The function contains five input variables, $x^1, \ldots, x^5$, and one output variable, y_i. The input variables are uniformly distributed over the interval of [0, 1]. To create drifting scenarios, one drifting dataset using the original Friedman's function was produced according to [50]. The dataset also has 2000 data and sudden concept drifts. The concept drift is localized in two distinct regions of the instance space. The first region with drift is defined with the conjunction of inequalities $R_1 = \{x_2 < 0.3 \wedge x_3 < 0.3 \wedge x_4 > 0.7 \wedge x_5 < 0.3\}$, and the second region is defined with the conjunction of inequalities $R_1 = \{x_2 > 0.7 \wedge x_3 > 0.7 \wedge x_4 < 0.3 \wedge x_5 < 0.7\}$. Let M denote the size of the dataset. There are three points of abrupt change in the training dataset, the first one at $(1/4)M$ data, the second one at $(1/2)M$ data, and the third at $(3/4)M$ data. The complete description of these drifting data can be found in [50].

Fig. 3 shows the results of our comparison based on the drifting hyperplane dataset. From Fig. 3, we reach a similar conclusion to the first experiment, in general. For example, in most time windows that have no sudden drift, such as tw_1 ([0-200]), tw_2 ([200-400]), tw_7 ([1200-1400]), and tw_{10} ([1800-200]), the RMSE obtained by each online SVR algorithm shows a convergent trend as the training data increases. However, unlike the previous experiment, when sudden drift occurs, that is, $M = 500$ and $M = 1500$, the RMSE for every online SVR-based algorithm shows a more dramatic and rapid upward trend. Therefore, this also illustrates that sudden drift has a more serious impact on performance than incremental drift.

The last is a dataset with gradual drift. In this dataset, there are also four concepts with a function. The function contains three input variables, $x^1, \ldots, x^3 \in [0, 1]$, and one output variable, y_i. We also simulate a window (which includes 200 instances) of change where the probability of a given instance belonging to the current concept of the new concept is governed by a sigmoid function. Basically, at the start of the window, the probability that one instance is drawn from concept 1 is higher, while at the end of the window, the probability of it being drawn from concept 2 increases, after the window is over all instances will be drawn from concept 2

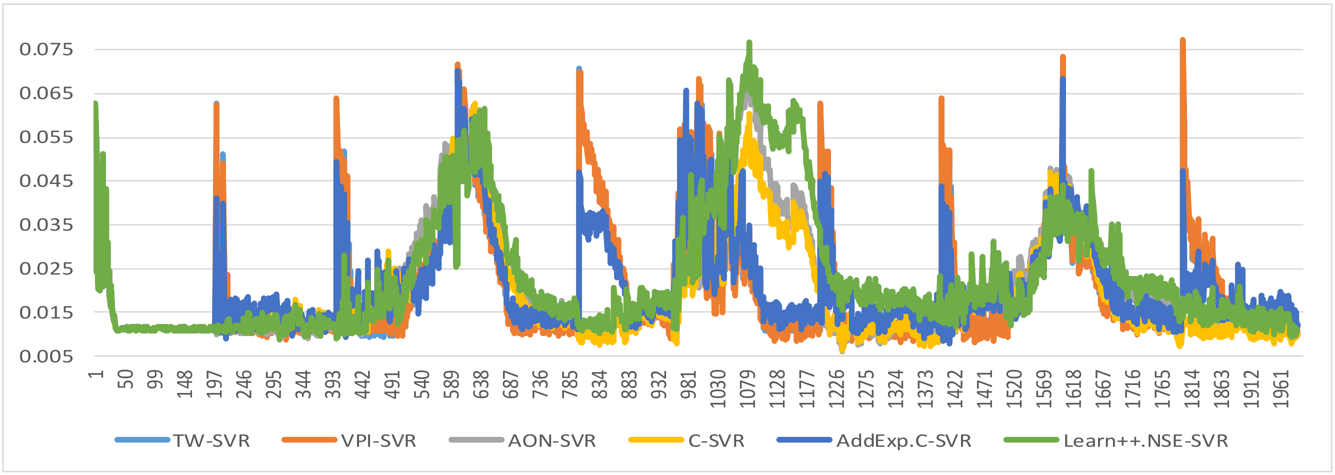

Fig. 4. Results on the gradual drifting dataset (the x-axis represents the size of instances, and the y-axis represents RMSE).

(the "new concept" is now stable). In this example, there are three gradual drifts centered around instances 500, 1000, and 1500, respectively.

Fig. 4 shows the results of our comparison based on the gradual drifting dataset. From Fig. 4, we still reach a similar conclusion to the first and second experiments, in general. However, unlike the previous two experiments, when the gradual drift window arrives, that is, [400-600] and [1400-1600], the RMSE for every online SVR-based algorithm shows a more rapid upward trend in [500-600] and [1500-1600] than in [400-500] and [1500-1600].

In summary, these results from the aforementioned three experiments show that our proposed C-SVR achieves more stable and better results than the other online SVR-based algorithms for nonstationary streaming data regression. The reason is C-SVR learns $f_i(X)$ which is dependent on $f_{i-1}(X)$. Hence, C-SVR is not like AON-SVR, VPI-SVR, TW-SVR, that is, the learned knowledge is discarded in a new tw_{i+1} with or without concept drift in tw_i. In contrast, when there are no concept drifts in tw_i, the learned knowledge can be saved in C-SVR, thereby the learning of $f_i(X)$ does not need to start from an initial state and can make an accurate prediction as soon as possible. On the other hand, C-SVR is not like AddEXP.C-SVR, Learn++.NSE-SVR, that is, outdated $f_i(X)$ cannot be discarded in time. In C-SVR, if there are concept drifts in tw_i, the learned knowledge will be discarded, therefore the learning of $f_i(X)$ is not affected by outdated knowledge, thereby reducing the impact of outdated models on the prediction results.

Next, we discuss the impact of parameters on the performance of each online SVR algorithm. Two parameters C and ε need to be set in each online SVR algorithm. The impact of these two parameters on performance is discussed in former publications [14], [17]. Therefore, there is no need to discuss the impact of these two parameters on each online SVR-based algorithm here. Each online SVR-based algorithm uses the same parameters C and ε. However, from the above two experiments' results, we can see that the performance of each algorithm in different time windows is different. Therefore, to further illustrate the advantage of our proposed C-SVR, we designed the third experiment to discuss the impact of the time windows' size on the performance of each algorithm. In this experiment, we set the size of the time window to 50, 100, 200, 400, and 500. We then compared the RMSE and the learning time of each algorithm based on time windows of different sizes. Fig. 5 shows the comparative results.

Fig. 5(a)–(c) shows that the average RMSE obtained by TW-SVR, VPI-SVR, and AddExp.C-SVR, in three datasets with concept drift, are significantly different if using time windows of different sizes. However, when the new time window arrives at exactly the time that each concept drift occurs, the average RMSEs obtained by TW-SVR, VPI-SVR, and AddExp.C-SVR are smaller than when the concept drift does not occur when that time window arrives. In contrast, there is little difference between the average RMSE obtained by AON-SVR and Learn+.NSE-SVR and our proposed C-SVR when using time windows of different sizes. Of these, the RMSE obtained by AON-SVR and C-SVR increases slightly with the increase of the time window's size, while Learn+.NSE-SVR decreases slightly with the increase of the time window's size. Fig. 5(d)–(f) shows that the learning time of TW-SVR, VPI-SVR, AddExp.C-SVR, AON-SVR, and C-SVR increases with the increase of the time window's size. The reason for this is that the training data increases with the increase in the size of the time windows, which results in the increase of data that belongs to the three sets—S_S (Supporting set), E_S (Error set), and R_S (Remaining set). Therefore, when learning new data, the data in the three sets that need to be adjusted also increases, thereby increasing learning time. AON-SVR needs to choose a datum to discard, so the learning time is longer than other online SVR algorithms. Learn++. NSE-SVR only relearns a new function when concept drift occurs, so its learning time is less affected by the size of the time window, but when the size of the time window is less than 500, the learning time of Learn++. NSE-SVR is much longer than the other online SVR-based algorithms.

We also carry out comparative experiments using online SVR-based algorithms and other types of online regression

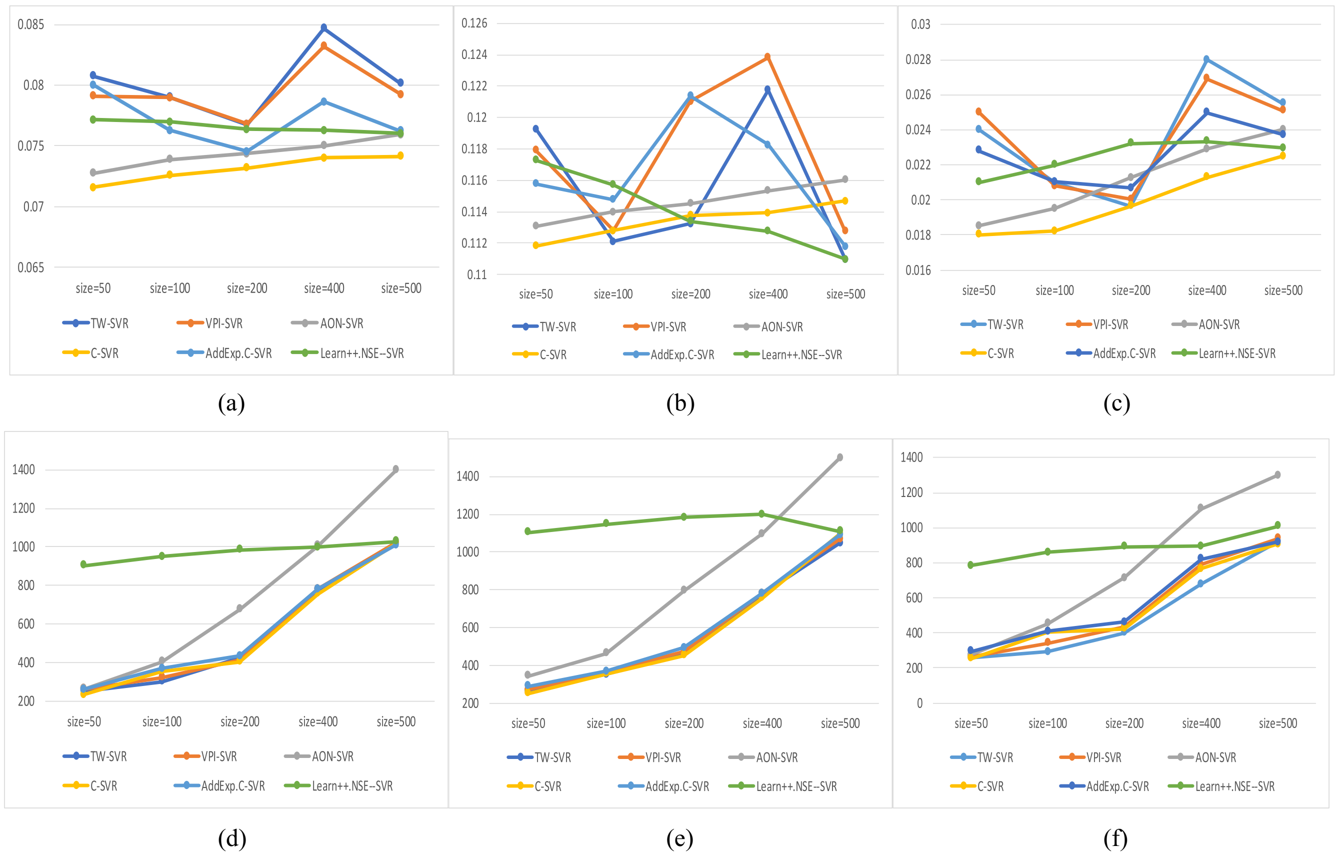

Fig. 5. The size of time window. (a) RMSE on the drifting hyperplane dataset. (b) RMSE on the drifting Friedman's dataset. (c) RMSE on the gradual drift dataset. (d) Time on the drifting hyperplane dataset. (e) Time on the drifting Friedman's dataset. (f) Time on the gradual drift dataset.

TABLE I
COMPARISON RESULTS ON SYNTHETIC DATASETS (RMSE)

	Hyperplane	Friedman	Gradual drifting datasets	Average-Rank
FMITDD	4.42E-01 (9)	2.63E-01 (8)	9.82E-02 (9)	8.67
ORTO	10.27E-01 (11)	8.12E-01 (11)	3.08E-01 (11)	11
ARF-Reg	3.88E-01 (8)	4.67E-01 (10)	8.41E-02 (8)	8.67
AMRules	4.48E-01 (10)	3.12E-01 (9)	1.10E-01 (10)	9.67
I-SVR	2.31E-01 (7)	14.82E-01 (7)	3.74E-02 (7)	7.00
TW-SVR	8.01E-02 (6)	11.49E-02 (5)	2.36E-02 (6)	5.67
VPI-SVR	7.98E-02 (5)	11.78E-02 (6)	2.35E-02 (5)	5.33
AON-SVR	7.49E-02 (2)	11.47E-02 (3)	2.12E-02 (2)	2.33
AddExp.C-SVR	7.54E-02 (3)	11.67E-02 (4)	2.26E-02 (4)	3.67
Learn++.NSE-SVR	7.61E-02 (4)	11.32-02 (2)	2.25E-02 (3)	3.00
C-SVR	7.38E-02 (1)	11.29E-02 (1)	1.99E-02 (1)	1.00

algorithms. An incremental SVR algorithm, called I-SVR, is included to demonstrate the advantages of a forgetting mechanism. I-SVR is without a forgetting mechanism and uses an incremental solution as in [14] to solve the QPP of a classical ε-SVR. Online SVR-based algorithms with a forgetting mechanism include TW-SVR, VPI-SVR, AON-SVR, AddExp.C-SVR, Learn++.NSE-SVR, and our proposed algorithm C-SVR. For each dataset, we set time windows of different sizes; the details are given in Table I. Based on the size of the time window, we used a parameter optimization method [52] to find the approximate values of the parameters C and ε of all online SVR-based algorithms. For some parameters that need be set to AddExp.C and Learn++.NSE, we used the same parameters as those used in the artificial datasets. Other types of online regression algorithms include FIMT-DD [53], ORTO [57], ARF-Reg [58], and AMRules [55] which are implemented by the MOA software [59].

TABLE II
DATASETS DETAILS

	Name	Instances	Attributes	Time-window size
Artificial	Hyperplane (incremental drift)	2000	10	200
	Friedman (sudden drift)	2000	5	200
	Gradual drift dataset	2000	3	200
Real-world	CCPP	9568	5	300
	Bikes	17389	16	500
	Cpusmall	8192	12	400
	House	20640	8	1000
	Solar	32686	6	1000

TABLE III
COMPARISON RESULTS ON REAL-WORLD DATASETS

	CCPP	Bikes	Cpusmall	House	Solar	Average-Rank
FMITDD	3.57E+00 (9)	2.57E+00 (2)	**1.03E+00 (1)**	5.54E+04 (3)	1.14E+02 (9)	4.8
ORTO	4.53E+02 (11)	5.93E+01 (11)	9.31E+01 (11)	9.42E+04 (10)	2.25E+02 (11)	10.8
ARF-Reg	3.72E+00 (10)	**7.29E-01 (1)**	1.12E+00 (2)	10.07E+04 (11)	9.47E+01 (4)	5.6
AMRules	3.42E+00 (8)	2.92E+00 (3)	1.31E+00 (3)	**4.72E+04 (1)**	9.52E+01 (5)	4
I-SVR	**3.04E+00 (1)**	6.30E+00 (10)	1.92E+00 (9)	7.19E+04 (9)	1.11E+02 (10)	7.8
TW-SVR	3.36E+00 (6)	5.17E+00 (9)	1.97E+00 (10)	6.01E+04 (6)	1.08E+01 (6)	7.4
VPI-SVR	3.32E+00 (7)	5.02E+00 (8)	1.83E+00(8)	6.06E+04 (7)	1.04E+01 (7)	7.4
AON-SVR	3.12E+00 (2)	4.29E+00 (6)	1.74E+00 (7)	6.28E+04 (8)	1.18E+02 (8)	6.2
AddExp.C-SVR	3.29E+00 (5)	4.30E+00 (7)	1.59E+00 (6)	5.87E+04 (5)	9.46E+01 (3)	5.2
Learn++.NSE-SVR	3.24E+00 (4)	3.22E+00 (4)	1.51E+00 (5)	5.82E+04 (4)	9.42E+01 (2)	3.8
C-SVR	3.18E+00 (3)	3.65E+00 (5)	1.46E+00 (4)	4.83E+04 (2)	**9.40E+01 (1)**	**3**

Table I shows the comparison results for every algorithm in terms of RMSE for different datasets. The results demonstrate that our proposed C-SVR obtains the best performance.

C. Real-World Datasets

In this section, we present the experiment results from six real-world regression datasets. The datasets were selected from different applications with a wide range of data sizes and dimensionality. Table II lists the details for each dataset.

The datasets are described as follows: the CCPP and Bikes datasets are taken from the UCI repository (http://archive.ics.uci.edu/ml/). The CCPP dataset is collected from a combined cycle power plant over six years (2006–2011), and the work in [51] proved that does not have any drift. The Bikes dataset contains daily counts of rented bicycles from the bicycle rental company *Capital-Bikeshare* in Washington, DC, USA, with weather and seasonal information. The Cpusmall dataset is taken from the Delve datasets (http://www.cs.toronto.edu/~delve/data/datasets.html), and the goal is to predict a computer system activity from system performance measures. The House dataset is a collection of all the housing block groups in California from the 1990 Census. A block group on average includes 1425.5 individuals living in a geographically compact area. The Solar dataset is provided by NASA and is taken from the HI-SEAS weather station in the period between Mission IV and Mission V. The data interval is roughly 5 min.

Table III shows the comparison results for every algorithm in terms of RMSE for different datasets. The results demonstrate that although I-SVR obtains the best performance on the CCPP datasets, that is, there is no clear concept drift in the datasets, the performance of other online regression algorithms with a forgetting mechanism is not much worse than that of I-SVR. In contrast, in the last four datasets, that is, there may be concept drift in the datasets, online regression algorithms with the forgetting mechanism significantly outperform I-SVR. Therefore, the forgetting mechanism is an important key in solving the online nonstationary regression problem. Next, to compare the results obtained by the online SVR-based algorithms and our proposed C-SVR, we can see that C-SVR achieved a better performance than the other online SVR-based algorithms on the last four algorithms, except for Learn++.NSE-SVR. The reason for this may be because the performance of Learn++.NSE-SVR is not impacted by the size of the time window, so Learn++.NSE-SVR achieves a better performance than C-SVR when the size of the time window has not been suitably set. However, for most of the datasets, C-SVR achieves the best performance when the size of the time window has been set to a multiple of 100. In addition, with an increasing amount of data being assigned to one of the three sets (S_S, E_S, and R_S) and no data to discard, the learning speed will become increasingly slower, until it is much slower than that of C-SVR. Next, to compare the results obtained by FIMT-DD, ORTO, ARF-Reg, AMRules, and our proposed C-SVR, we can see that C-SVR can also achieve comparative performance. However, the main advantage of the C-SVR is stability (has minimum average-rank). The reason for this is that the difference between incremental drift, gradual drift, and sudden drift was omitted in ORTO, FIMT-DD, ARF-Reg, and AMRules, but was considered in our proposed method.

V. Conclusion

It is well known that the classical ε-SVR can be transformed into an online-based SVR to handle regression problems with streaming data using an incremental solution to solve the QPP. However, with nonstationary streaming data, many online SVR algorithms have failed. Hence, in this article, we proposed C-SVR as a method for solving regression problems with nonstationary streaming data. C-SVR is based on a continuous learning strategy that learns a series of $f_i(X)$ in a series of tw_{i}. However, only one $f_i(X)$ for making a prediction is saved in the computer's memory. A new $f_i(X)$ is learned when each new tw_{i} arrives, and the last $f_{i-1}(X)$ that was learned in the last $tw_{\mathrm{i}-1}$ is discarded. A unique difference with our approach is an added similarity term to the QPP that makes the new $f_i(X)$ dependents on $f_{i-1}(X)$ by transferring some learned knowledge between time windows. The amount of transferred learned knowledge is determined by the extent of the drift as measured by the competency-based method. Like other SVRs designed to work with streaming data, C-SVR solves the QPP in an online manner. Experiments indicate that C-SVR achieves better performance than several online SVR algorithms, especially for datasets with incremental concept drift.

In the future, we intend to look beyond fixed-size time windows with a similar TAI-SVR method that involves varying the size of the time windows to further improve the accuracy of C-SVR. The reason for this is the interval between the occurrences of two concept drifts is not necessarily fixed, so using time windows of varying size is helpful to determine the degree of concept drift more accurately.

References

[1] J. M. Moreira, C. Soares, A. M. Jorge, and J. F. de Sousa, "Ensemble approaches for regression: A survey," *ACM Comput. Surveys*, vol. 45, no. 1, pp. 1–10, 2012.

[2] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Neural Information Processing Systems 9*, M. Mozer, J. Jordan, and T. Petscbe, Eds. Cambridge, MA, USA: MIT Press, 1997, pp. 155–161.

[3] A. J. Smola, B. Sch, and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[4] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," in *Advances in Kernel Method: Support Vector Learning*, Scholkopf, Burges, and Smola, Eds. Cambridge, MA, USA: MIT Press, 1998, pp. 185–208.

[5] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to the SMO algorithm for SVM regression," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1188–1193, Sep. 2000.

[6] S. Poularakis and I. Katsavounidis, "Low-complexity hand gesture recognition system for continuous streams of digits and letters," *IEEE Trans. Cybern.*, vol. 46, no. 9, pp. 2094–2108, Sep. 2016.

[7] X. Miao, Y. Liu, H. Zhao, and C. Li, "Distributed online one-class support vector machine for anomaly detection over networks," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1475–1488, Apr. 2019.

[8] S. Zhao, Y. Gao, G. Ding, and T. S. Chua, "Real-time multimedia social event detection in microblog," *IEEE Trans. Cybern.*, vol. 48, no. 11, pp. 3218–3231, Nov. 2018.

[9] J. Gama and M. M. Gaber, *Learning From Data Streams: Processing Techniques in Sensor Networks*. New York, NY, USA: Springer, 2007.

[10] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: Theory and practice," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 3, pp. 515–528, May–Jun. 2003.

[11] S. Vijayakumar and S. Wu, "Sequential support vector classifiers and regression," in*Proc. Int. Conf. Soft Comput. (SOCO'99)*, Genoa, Italy, 1999, pp. 610–619.

[12] Y. Engel, S. Mannor, and R. Meir, "Sparse online greedy support vector regression," in *Proc. 13th Eur. Conf. Mach. Learn.*, 2002, pp. 84–96.

[13] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[14] J. Ma, J. Theiler, and S. Perkins, "Accurate on-line support vector regression," *Neural Comput.*, vol. 15, no. 11, pp. 2683–2703, 2003.

[15] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, MA, USA: MIT Press, 2001, pp. 409–415.

[16] W. Karush, "Minima of functions of several variables with inequalities as side constraints," M.S. thesis, Dept. Math., Univ. Chicago, Chicago, IL, USA, 1939.

[17] B. Gu, V. S. Sheng, Z. Wang, D. Ho, S. Osman, and S. Li, "Incremental learning for ν-support vector regression," *Neural Netw.*, vol. 67, pp. 140–150, Jul. 2015.

[18] B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Comput.*, vol. 12, no. 5, pp. 1207–1245, 2000.

[19] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training," *J. Mach. Learn. Res.*, vol. 13, pp. 3103–3131, Oct. 2012.

[20] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 7, pp. 1403–1416, Jul. 2015.

[21] H. Yu, J. Lu, and G. Zhang, "An online robust support vector regression for data streams," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 12, 2020, doi: 10.1109/TKDE.2020.2979967.

[22] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surveys*, vol. 46, no. 4, pp. 1–37, 2014.

[23] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[24] H. Yu, J. Lu, and G. Zhang, "Online topology learning by a Gaussian membership-based self-organizing incremental neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, early aceess, Nov. 13, 2019, doi: 10.1109/TNNLS.2019.2947658.

[25] G. Carpenter and S. Grossberg, "The ART of adaptive pattern recognition by a self-organizing neural network," *IEEE Comput.*, vol. 21, no. 3, pp. 77–88, Mar. 1988.

[26] J. Liu and E. Zio, "An adaptive online learning approach for support vector regression: Online-SVR-FID," *Mech. Syst. Signal Process.*, vols. 76–77, pp. 796–809, Aug. 2016.

[27] O. A. Omitaomu, M. K. Jeong, and A. B. Badiru, "Online support vector regression with varying parameters for time-dependent data," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 41, no. 1, pp. 191–197, Jan. 2011.

[28] R. Klinkenberg and T. Joachims, "Detection concept drift with support vector machines," in *Proc. 7th Int. Conf. Mach. Learn. (ICML)*, 2000, pp. 487–494.

[29] W. Kim, J. Park, J. Yoo, H. Kim, and C. G. Park, "Target localization using ensemble support vector regression in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 4, pp. 1189–1198, Aug. 2013.

[30] J. Liu, V. Vitelli, E. Zio, and R. Seraoui, "A novel dynamic-weighted probabilistic support vector regression-based ensemble for prognostics of time series data," *IEEE Trans. Rel.*, vol. 64, no. 4, pp. 1203–1213, Dec. 2015.

[31] J. Liu and E. Zio, "A SVR-based ensemble approach for drifting data streams with recurring patterns," *Appl. Soft Comput. J.*, vol. 47, pp. 553–564, Oct. 2016.

[32] C. Fahy, S. Yang, and M. Gongora, "Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2215–2228, Jun. 2019.

[33] Y. Song, J. Lu, H. Lu, and G. Zhang, "Fuzzy clustering-based adaptive regression for drifting data streams," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 3, pp. 544–557, Mar. 2020.

[34] M. J. Hosseini, A. Gholipour, and H. Beigy, "An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams," *Knowl. Inf. Syst.*, vol. 46, no. 3, pp. 567–597, 2016.

[35] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2017, pp. 4655–4665.

[36] B. Schölkopf and A. Smola, *Learning With Kernels—Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[37] K.-R. Müller, S. Mika, G. Rätsch, and K. Tsuda, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 181–201, Mar. 2001.

[38] R. Rockafellar, "Augmented lagrange multiplier functions and duality in nonconvex programming," *SIAM J. Control*, vol. 12, pp. 268–285, May 1974.

[39] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, Oct. 2013.

[40] I. Žliobaite and B. Gabrys, "Adaptive preprocessing for streaming data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 309–321, Feb. 2014.

[41] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models," *Artif. Intell.*, vol. 209, pp. 11–28, Apr. 2014.

[42] F. Dong, J. Lu, K. Li, and G. Zhang, "Concept drift region identification via competence-based discrepancy distribution estimation," in *Proc. 12th Int. Conf. Intell. Syst. Knowl. Eng. (ISKE)*, vol. 47. Nanjing, China, 2017, pp. 1–7.

[43] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi, "Test of page-hinckley, an approach for fault detection in an agro-alimentary production system," in *Proc. 5th Asian Control Conf.*, vol. 2. Melbourne, VIC, Australia 2004, pp. 815–818

[44] G. Bennett, "Probability inequalities for the sum of independent random variables," *J. Amer. Stat. Assoc.*, vol. 57, pp. 33–45, Mar. 1962.

[45] J. Z. Kolter and M. A. Maloof, "Using additive expert ensembles to cope with concept drift," in *Proc. Int. Conf. Mech. Learn. (ICML'05)*, Bonn, Germany, 2005, pp. 449–456.

[46] R. Elwell and R. Polikar, "Incremental learning of concept drift in non-stationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[47] D. P. Solomatine, and D. L. Shrestha, "Experiments with AdaBoost.RT, an Improved Boosting Scheme for Regression," *Neural Comput.*, vol. 18, no. 7, pp. 1678–1710, 2006.

[48] A. Shaker and E. Hullermeier, "IBLStreams: A system for instance-based classification and regression on data streams," *Evol. Syst.*, vol. 3, no. 4, pp. 235–249, 2012.

[49] J. H. Friedman, "Multivariate adaptive regression splines," *Ann. Stat.*, vol. 19, no. 1, pp. 1–67, 1991.

[50] E. Ikonomovska, "Algorithms for learning regression trees and ensembles on evolving data streams," Ph.D. dissertation, Dept. Doctoral, Jozef Stefan Int. Postgraduate School, Ljubljana, Slovenia, Oct. 2012.

[51] L.-Y. Wang, C. Park, K. Yeon, and H. Choi, "Tracking concept drift using a constrained penalized regression combiner," *Comput. Stat. Data Anal.*, vol. 108, pp. 52–69, Apr. 2017.

[52] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 281–305, Feb. 2012.

[53] E. Ikonomovska, J. Gama, and S. Dzeroski, "Learning model trees from evolving data streams," *Data Min. Knowl. Discover.*, vol. 23, no. 1, pp. 128–168, 2011.

[54] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discover. Data Min.*, 2000, pp. 71–80.

[55] E. Almeida, C. Ferreira, and J. Gama, "Adaptive model rules from data streams," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discover. Databases (ECML-PKDD'13)*, 2013, pp. 480–492.

[56] T. Zhai, Y. Gao, H. Wang, and L. Cao, "Classification of high-dimensional evolving data streams via a resource-efficient online ensemble," *Data Min. Knowl. Discover.*, vol. 31, no. 5, pp. 1242–1265, 2017.

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

[58] H. M. Gomes, J. P. Barddal, L. E. B. Ferreira, and A. Bifet, "Adaptive random forests for data stream regression," in *Proc. 26th Eur. Symp. Artif. Neural Netw. (ESANN)* Bruges, Belgium, Apr. 2018, pp. 267–272.

[59] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, Aug. 2010.

[60] H. Yu, J. Lu, J. Xu, and G. Zhang, "A hybrid incremental regression neural network for uncertain data streams," in *Proc. Int. Joint Conf. Neural Netw.*, Budapest, Hungary, 2019, pp. 1–8.

Hang Yu is currently pursuing the Ph.D. degree with the Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW, Australia.

He is a member of the Decision Systems and e-Service Intelligence Research Laboratory, Centre for Artificial Intelligence, University of Technology Sydney. His research interests include streaming data mining, concept drift, and fuzzy systems.

Jie Lu (Fellow, IEEE) received the Ph.D. degree from the Curtin University of Technology, Perth, WA, Australia, in 2000.

She is a Distinguished Professor and the Director of the Centre for Artificial Intelligence, University of Technology Sydney, Sydney, NSW, Australia. She has published six research books and over 450 papers in refereed journals and conference proceedings. She has won over 20 ARC Laureate, ARC Discovery Projects, and government and industry projects. Her main research interests are in the areas of fuzzy transfer learning, concept drift, decision support systems, and recommender systems.

Dr. Lu has received various awards, such as the UTS Medal for Research and Teaching Integration in 2010, the UTS Medal for Research Excellence in 2019, the Computer Journal Wilkes Award in 2018, the IEEE TRANSACTIONS ON FUZZY SYSTEMS Outstanding Paper Award in 2019, and the Australian Most Innovative Engineer Award in 2019. She serves as the Editor-in-Chief for *Knowledge-Based Systems* (Elsevier) and the *International Journal of Computational Intelligence Systems*. She has delivered over 25 keynote speeches at international conferences and chaired 15 international conferences. She is an IFSA Fellow and an Australian Laureate Fellow.

Guangquan Zhang (Member, IEEE) received the Ph.D. degree in applied mathematics from the Curtin University of Technology, Perth, WA, Australia, in 2001.

He is an Associate Professor and the Director of the Decision Systems and e-Service Intelligent Research Laboratory, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW, Australia. He has authored four monographs, five textbooks, and 450 papers in *Artificial Intelligence*, *Machine Learning*, IEEE TRANSACTIONS ON FUZZY SYSTEMS, and other refereed journals and conference proceedings. His research interests include fuzzy machine learning, fuzzy optimization, and machine learning and data analytics.

Dr. Zhang was awarded an ARC QEII Fellowship in 2005. He has served as a member of the editorial boards of several international journals, as a guest editor of eight special issues for the IEEE TRANSACTIONS and other international journals and co-chaired several international conferences and workshops in the area of fuzzy decision making and knowledge engineering. He has won seven Australian Research Council (ARC) Discovery Project grants and many other research grants.