A Segment-based Drift Adaptation Method for Data Streams

Yiliao Song, *Student Member, IEEE*, Jie Lu, *Fellow, IEEE*, Anjin Liu, *Member, IEEE*, Haiyan Lu, *Senior Member, IEEE*, Guangquan Zhang

***Abstract*—In concept drift adaptation, we aim to design a blind or an informed strategy to update our best predictor for future data at each time point. However, existing informed drift adaptation methods need to wait for an entire batch of data to detect drift and then update the predictor (if drift is detected), which causes adaptation delay. To overcome the adaptation delay, we propose a sequentially updated statistic, called *drift-gradient* to quantify the increase of distributional discrepancy when every new instance arrives. Based on drift-gradient, a *segment-based drift adaptation*(SEGA) method is developed to online update our best predictor. Drift-gradient is defined on a segment in the training set. It can precisely quantify the increase of distributional discrepancy between the old segment and the newest segment when *only one* new instance is available at each time point. A lower value of drift-gradient on the old segment represents that the distribution of the new instance *is closer* to the distribution of the old segment. Based on the drift-gradient, SEGA retrains our best predictors with the segments that have the minimum drift-gradient when every new instance arrives. SEGA has been validated by extensive experiments on both synthetic and real-world, classification and regression data streams. The experimental results show that SEGA outperforms competitive blind and informed drift adaptation methods.**

***Index Terms*—drift adaptation, concept drift, online learning, data stream.**

I. Introduction

Data streams consists of infinite [1] and fast evolving data instances which arrive in a sequential way [2], [3].For a data stream, the newly arrived data instances may exhibit a different pattern from the previous data [4]. This phenomenon is called concept drift, where the terminology "concept" refers to the hidden patterns or the underlying distribution of data [5]. For example, the pattern of raining may change in different time period.

To solve the concept drift problem in a data stream, concept drift adaptation aims to design a *blind* or an *informed* strategy to update the best predictor for future data at each time point [6]. The adaptation method with a blind strategy is also categorised as "passive" approach and the adaptation method with an informed strategy categorised as "active" approach [7].

However, existing informed drift adaptation methods need to wait for an entire batch (time window) of data to detect drift and then update the predictor (if drift is detected), which causes adaptation delay. Specifically, the existing informed adaptation methods only use the result of drift detection process to assist adaptation [8]. They need a batch of data to determine whether drift occurs [9], [10], which results in the drift reaction delay.

To overcome the adaptation delay, we propose a sequentially updated statistic, called *drift-gradient*, and develop a *segment-based drift adaptation* (SEGA) method to update our best predictor when every new instance arrives. During the learning process, the training data is ordered by time and divided into several equal length segments, and a predictor is learned with the data in each segment. When a new instance arrives, we first update the drift-gradient on each segment in the training data. Based on the updated drift-gradient, SEGA retrains our best predictors with the segments that have the minimum drift-gradient.

Drift-gradient is to quantify the increase of *segmented symmetric degree* (SSD) when *only one* new instance is available at each time point. SSD is a new statistic proposed to measure the distributional discrepancy between an old segment and the newest segment in the training set. The advantage of drift-gradient is that it does *not* need to compute the value of SSD before and after the new instance arrives, and therefore can be sequentially updated with low computational cost. A lower value of drift-gradient on an old segment represents that the distribution of the new instance *is closer* to the distribution of this old segment. Therefore, the predictor trained with this segment should be a better predictor for the new instance.

SEGA has been validated by extensive experiments on both synthetic and real-world, classification and regression data streams. To our best knowledge, this is the first time that the drift adaptation method has been comprehensively tested on both continuous and discrete label variables. SEGA has been compared to 15 benchmarks (where 7 of them are adaptation methods for regression task, and 8 are for classification task) and validated on 12 synthetic data (where 6 of them are for the regression task and 10 are for the classification task) and 14 real-world data streams (where 7 of them are for the regression task and 7 are for the classification task). The experimental results show that SEGA outperforms competitive blind and informed drift adaptation methods.

The contributions of this paper includes three aspects:

- A *segmented symmetric degree* (SSD) is proposed to measure distributional discrepancy between old segments and the newest segment in the training set. SSD is better than a one-sided measurement when two distributions have different variances;

Y. Song, J. Lu, A. Liu, H. Lu and G. Zhang are with the Decision Systems and e-Service Intelligence Laboratory, Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: Yiliao.Song@student.uts.edu.au; Anjin.liu@uts.edu.au; Jie.Lu@uts.edu.au; Haiyan.Lu@uts.edu.au; Guangquan.Zhang@uts.edu.au.)

- A sequentially updated statistic, *drift-gradient* is proposed to quantify the **increase of SSD** (not SSD) when every new instance arrives *without* computing the value of SSD before and after the new instance arrives;
- An *online drift adaptation* method, SEGA is developed based on drift-gradient. SEGA can effectively overcome the adaptation delay issue in the existing informed drift adaptation methods.

The remainder of this paper will cover the following: related concept drift research is reviewed in Section II. Section III explains the principle and procedure of the proposed SEGA method. The experimental evaluations of the SEGA method are presented in Section IV, and finally, this paper concludes with a discussion of the future work in Section V.

II. Related Work

A. An overview of concept drift

The idea of concept drift is proposed, to enable machine learning predictors to be applicable in data streams with changing distributions [11]. Much of the existing research has validated the importance of solving the problem of concept drift problem, from theory to practice [9], [12], [13].

A widely-accepted definition of concept drift is presented in the form of probability distribution that $\exists t_d : p_{t_d+1}(\boldsymbol{X}, y) \neq p_{t_d}(\boldsymbol{X}, y)$ [14]. The changes of $p(y|\boldsymbol{X})$ are called real drift [15], [16] and the changes of $p(\boldsymbol{X})$ are virtual drift [17], [18]; Concept drift can occurs as four types: *sudden drift*, *incremental drift*, *gradual drift* and *reoccurring drift* [10]. This paper focuses on all four types of the real drift.

B. Concept drift adaptation

The occurrence of concept drift denotes a trained predictor is no longer suitable for predicting upcoming data, and the concept drift adaptation methods are designed to solve this problem [14], [19]. There are three basic requirements in drift adaptation: 1) **adaptation should be fast** [20]. For example, for a drift adaptation method, the adaptation process should be done in half an hour if the frequency of data is half an hour. 2) **adaptation should be robust** [21]. 3) **adaptation should also be applicable in a non-drift data stream** [22]. As it is uncertain whether drift would occur in a real-world data stream, the designed drift adaptation methods should also be to applied in a non-drifted data stream.

Existing drift adaptation is implemented by two strategies: *blind* adaptation strategy [23], [24] and *informed* adaptation strategy [14]. Blind adaptation strategies do not involve drift detection techniques such as evolving neural networks in [25], AUE2 [26], DWMIL [27] and DTEL [28]. The predictor could be ill-trained with the "always-adapt" mechanism in the blind strategy [22]. In informed adaptation, a drift detection technique is implemented help adaptation [29]. The drift detection can be implemented based on the error [30], [31] or a statistic [32], [33].

In existing informed adaptation methods, the drift detection technique is used to output a binary variable of "drift is detected" or "drift is not detected", which has inevitable time delay [10]. In [34], [35], the PH-test statistic is updated when a new instance arrives by adding a fixed parameter [34] or the value of a fixed function of parameters [35]. Clearly, this forgetting mechanism assumes that the drift has a regular pattern, which is not always true in the real applications.

In this paper, we propose a statistic SSD which can be used to detect drift. Instead of directly using SSD to identify whether drift occurs, we propose a sequentially updated statistic, called *drift-gradient* to present the changes of SSD without computing the value of SSD before and after a new instance arrives. Drift-gradient can be used to select the most appropriate segments in the current training set to make adaption.

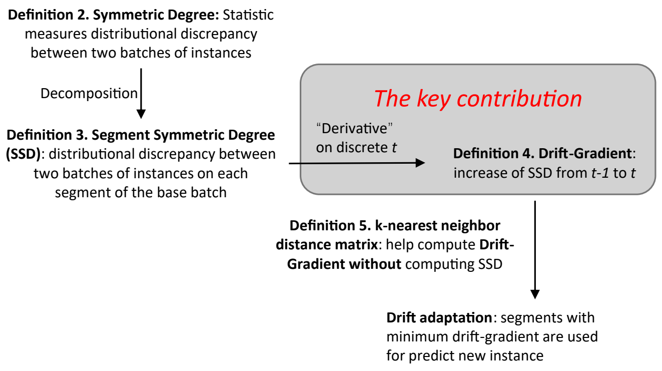

Definition 2. Symmetric Degree: Statistic measures distributional discrepancy between two batches of instances
Decomposition
Definition 3. Segment Symmetric Degree (SSD): distributional discrepancy between two batches of instances on each segment of the base batch
The key contribution
"Derivative" on discrete *t*
Definition 4. Drift-Gradient: increase of SSD from *t-1* to *t*
Definition 5. k-nearest neighbor distance matrix: help compute **Drift-Gradient without** computing SSD
Drift adaptation: segments with minimum drift-gradient are used for predict new instance

Fig. 1. Relationship between the definitions proposed in this paper.

III. Methodology

This paper proposes a sequentially-updated statistic, called *drift-gradient* and based on it, a *segment-based adaptation*(SEGA) method is developed to update the predictor when every new instance arrives. Drift-gradient is to quantify the increase of *segmented symmetric degree* (SSD) when *only one* new instance is available at each time point. SSD is a new statistic that can measure the distributional discrepancy between old segments and the newest segment. The relationship between the proposed new definitions is presented in Fig. 1.

In this section, we will firstly introduce the problem setting, then introduce SSD, drift-gradient and finally SEGA. Basic notations and the problem setting of drift adaptation are outlined in section III-A. The proposed SSD is presented in section III-B. Section III-C explains how to update drift-gradient, and finally, SEGA is presented in section III-D.

A. Problem setting

To mimic the characteristic of data stream where data instances are observed in a sequential way, the learning and evaluation process for data stream is repeatedly activated when new data instances are observed. Specifically, a trained predictor is applied to predict the label value of the new data instance before the true label is obtained. After the true label is obtained, an evaluation process is activated and the instance is included to retrain the existing predictor.

We refer to the concepts and notation of concept drift in [28] and [16]. Given a data stream $D_t = \{(\boldsymbol{X}_t, y_t)\,|t = 1, ...\infty\}$, generated from distribution $\mathcal{P}_t$. $p_t(\boldsymbol{X}, y)$ denotes the probability of $\mathcal{P}_t$ in a discrete case or the probability density function (*pdf*) of $\mathcal{P}_t$ in a continuous case. Concept drift is defined as $\exists t_d : p_{t_d+1}(\boldsymbol{X}, y) \neq p_{t_d}(\boldsymbol{X}, y)$.

So far, the widely accepted definition of concept drift has highlighted the characteristics of *drift*, but has not explained the meaning of "*concept*". When studying the problem of concept drift, the term "concept" is used to represent the hidden data patterns such as the probability distributions and relationships between X and y. Concept drift is caused by the hidden context, rather than stochastic disturbances. Unlike outliers, a concept will last for a period after it shows, rather than existing momentarily. To present this characteristics of concept, a constraint is added to the current definition of concept drift as is presented in Definition 1.

Definition 1: **Concept Drift** occurs in a data stream if $\exists t_{d^{(i)}}$ that

$$\begin{cases} p_{t+1}(\boldsymbol{X}, y) \neq p_t(\boldsymbol{X}, y), \text{ for } t = t_{d^{(i)}} \\ p_{t+1}(\boldsymbol{X}, y) = p_t(\boldsymbol{X}, y), \text{ for } t \in \left[t_{d^{(i)}+\tau_i}, t_{d^{(i+1)}}\right) \end{cases} \quad (1)$$

where $\forall i,\ t_{d^{(i+1)}} - t_{d^{(i)}} > 1$, $t \in \mathbb{Z}^+$ presents the time point, $d^{(i)}$ is an order statistics denoting the i^{th} drifted time point, and $1 < \tau_i < t_{d^{(i+1)}} - t_{d^{(i)}}$ is specifically for the occurrence of incremental drift.

Remark. In this definition, a data stream contains concept drift if the data pattern changes at least once, namely $\{t_{d^{(i)}}\} \neq \emptyset$ that $p_{t+1}(\boldsymbol{X}, y) \neq p_t(\boldsymbol{X}, y)$, for $t = t_{d^{(i)}}$; in addition, the changed pattern is not ephemeral, but will last for a period (at least last for two time points), which is manifested by $\forall i,\ t_{d^{(i+1)}} - t_{d^{(i)}} > 1$. The pattern stays the same in this period that $p_{t+1}(\boldsymbol{X}, y) = p_t(\boldsymbol{X}, y)$, for $t \in \left(t_{d^{(i)}+\tau_i}, t_{d^{(i+1)}}\right)$; here $\tau_i = 1$ when the drift occurs suddenly while $\tau_i > 1$ when the drift occurs incrementally in the period of $\left(t_{d^{(i)}+1}, t_{d^{(i)}+\tau_i}\right)$. **All drift adaptation methods are at least one-instance delayed. Without the constraint that a new pattern will retain for a period, the adaptation is invalid in principle.**

According to Definition 1, a concept will exists for at least a time period of τ once it appears, and the occurrence of concept drift at t_d means the end of one concept. During the time period of one specific concept, the learning goal is to obtain a predictor H_c for $p_c(\boldsymbol{X}, y)$, which can be denoted as

$$H_c = \underset{h \in \mathcal{H}}{\arg\min} \mathbb{E}_{(\boldsymbol{X}, y) \sim p_c}[\ell(h(\boldsymbol{X}), y)], \quad (2)$$

where $\mathcal{H}$ is the hypothesis set, $\ell : \mathbb{R}^1 \times \mathbb{R}^1 \to \mathbb{R}_+$ is the loss function used to measure the magnitude of error. The goal of the learning process on the whole data stream is to minimize the loss over all concepts, which is to obtain the best predictors $h_1, h_2, ..., h_c, ...$ at each time point in (3).

$$\underset{h_1, h_2, ..., h_c, ...}{\arg\min} \sum_c \mathbb{E}_{(\boldsymbol{X}, y) \sim p_c}\left[\ell\left(h_c(\boldsymbol{X}), y\right)\right]. \quad (3)$$

In this paper, $h_1, h_2, ..., h_c, ...$ is determined by drift-gradient. Drift-gradient is to measure the increase of a proposed statistic SSD. In the next two subsections, we will introduce SSD and explain how to compute drift-gradient without computing the values of SSD.

B. The segmented symmetric degree (SSD)

SSD is the decomposition of a symmetric degree (SD) on segments. SD defined in Definition 2 is to measure the distributional discrepancy between two samples of data.

In this subsection, we use the following notations.

- P, P_1, P_2, Q: data samples
- N_P, N_{P_1}, N_{P_2}, N_Q: sample size
- p, p_1, p_2, q: probability distributions
- $\mathcal{K}_i(k)$: the set of k nearest neighbors of an instance i
- $N_{i,S}(k) = |\mathcal{K}_i(k) \cap S|$: how many k nearest neighbors of i are in S

In this paper, we use a kNN search to determine the neighbors for each instance. For both regression and classification tasks, we first normalize each variable (including the target variable) into $[0, 1]$ by using the min-max normalization. After that, the distance is computed by the Euclidean distance. However, using Euclidean distance denotes that we assume $(\boldsymbol{X}_t, y_t) \in \mathbb{R}^{d+1}$ which is not always true in real application, especially for the classification task. The kNN could also be determined by other distance, which can further improve the method.

Definition 2: **Symmetric Degree.** Given two data samples P and Q with p and q its distribution, the symmetric degree $\bar{d}_{P,Q}(k)$ is defined as the average density difference of P's and Q's k-nearest neighbors.

$$\begin{aligned} \bar{d}_{P,Q}(k) = & \frac{1}{N_P} \sum_{u \in P} \left(\frac{N_{u,P}(k)}{N_p} - \frac{N_{u,Q}(k)}{N_Q} \right) + \\ & \frac{1}{N_Q} \sum_{v \in Q} \left(\frac{N_{v,Q}(k)}{N_Q} - \frac{N_{v,P}(k)}{N_P} \right), \end{aligned} \quad (4)$$

where for any u, v, $N_{u,P}(k) + N_{u,Q}(k) = k$ and $N_{v,Q}(k) + N_{v,P}(k) = k$.

If there are more than one candidates for the kth-nearest neighbor of u, the kth-nearest neighbor is randomly chosen from the candidates in P. Similarly for u, v's kth-nearest neighbor is determined from the candidates in Q if there are more than one candidate. The first term in the summation of Definition 2, measures the average density difference over all of P's neighborhoods and the second term measures the average density difference over Q's neighborhoods. Clearly, the proposed $\bar{d}_{P,Q}(k)$ is symmetric on these two samples, given the same k, namely $\bar{d}_{P,Q}(k) = \bar{d}_{Q,P}(k)$. A larger absolute value of SD, means that P is more likely to be different from Q. As SD is a summation-based statistic, it converges to a normal distribution according to the central limit theorem, when N_P and N_Q approaches infinity. Similar proof can be found in our previous studies [16], [36].

Next, a simple example in Fig. 2 is introduced to show why the symmetric measurement is better than the one-side measurement. The one-sided measurement, such as the measurements in [16], [36], only counts the density difference on either P or Q's neighborhoods rather than both of them. In Fig. 2, the red dots represents sample P and the blue dots represents sample Q. According to the distribution of P and Q, they have the same expectation but P has larger variance. If the density difference is restricted to P's neighborhoods, as shown in subplot (a). When $k = 2$, the neighborhoods of P

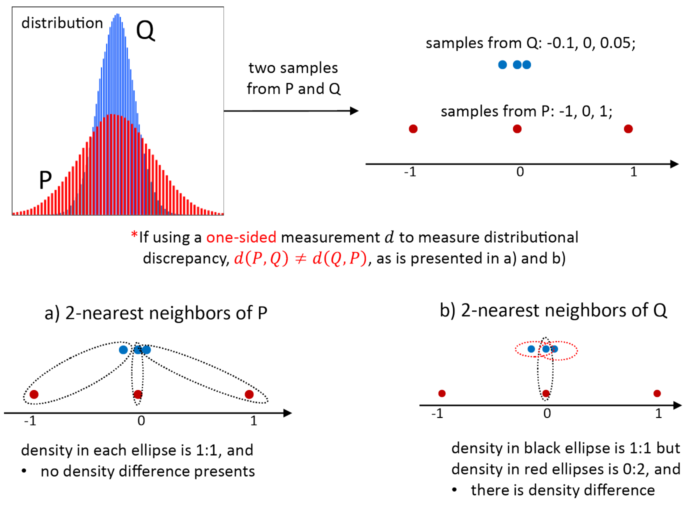

distribution
Q
P
two samples from P and Q
samples from Q: -0.1, 0, 0.05;
samples from P: -1, 0, 1;
-1
0
1
*If using a one-sided measurement d to measure distributional discrepancy, $d(P,Q) \neq d(Q,P)$, as is presented in a) and b)
a) 2-nearest neighbors of P
b) 2-nearest neighbors of Q
density in each ellipse is 1:1, and
• no density difference presents
density in black ellipse is 1:1 but density in red ellipses is 0:2, and
• there is density difference

Fig. 2. An example to show the drawback of a one-sided measurement. There is no density difference if neighbors of P are considered. However, density difference exists if neighbors of Q are considered.

is framed by the black dotted ellipse and there is no density difference in any ellipse. If the density difference is restricted to Q's neighborhoods in subplot (b), the neighborhoods of Q is framed up by three ellipses. In the black ellipse, there is no density difference between the blue and red dots. However, in the red ellipses, they all contain blue dots. The density of the blue dots is 1 and the density of the red dots is 0, giving a density difference of 1. Therefore, if a one-sided measurement of density difference is applied based on the sample that has larger variance, such as is in subplot (a), it is invalid to reflect the variance discrepancy between two samples.

Given SD able to measure the difference between two samples, let the current training data be sample P and the newest batch of data instances be the sample Q, where $N_P \gg N_Q$, SD can be used to test whether Q's distribution is different from P.

SSD is the decomposition of SD which can consider the distributional discrepancy on each segment of the training data. We firstly explain the decomposition of SD in the case of two segments. Given P_1 and P_2 two absolute complements in P that $P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$, SD can be rewritten as (5). Detailed derivation from SD to SSD can be found in Appendix.

$$\begin{aligned}\bar{d}_{P,Q}(k) =& \frac{1}{N_P}\sum_{u\in P_1}\left(\frac{N_{u,P}(k)}{N_P}-\frac{N_{u,Q}(k)}{N_Q}\right)+\\ &\frac{1}{N_P}\sum_{u\in P_2}\left(\frac{N_{u,P}(k)}{N_P}-\frac{N_{u,Q}(k)}{N_Q}\right)+\\ &\frac{1}{N_Q}\sum_{v\in Q}\left(\frac{N_{v,Q}(k)}{N_Q}-\frac{N_{v,P}(k)}{N_P}\right).\end{aligned} \tag{5}$$

Given $ssd_{P_1,Q}(k)$ and $ssd_{P_2,Q}(k)$ in (6) and (7), $\bar{d}_{P,Q}(k) = ssd_{P_1,Q}(k) + ssd_{P_2,Q}(k)$.

$$\begin{aligned}ssd_{P_1,Q}(k) =& \frac{1}{N_P}\sum_{u\in P_1}\left(\frac{k-N_{u,Q}(k)}{N_P}-\frac{N_{u,Q}(k)}{N_Q}\right)-\\ &\frac{1}{N_Q}\sum_{v\in Q}\frac{N_{v,P_1}(k)}{N_P}+\frac{1}{2N_Q}\sum_{v\in Q}\frac{N_{v,Q}(k)}{N_Q}.\end{aligned} \tag{6}$$

$$\begin{aligned}ssd_{P_2,Q}(k) =& \frac{1}{N_P}\sum_{u\in P_2}\left(\frac{k-N_{u,Q}(k)}{N_P}-\frac{N_{u,Q}(k)}{N_Q}\right)-\\ &\frac{1}{N_Q}\sum_{v\in Q}\frac{N_{v,P_2}(k)}{N_P}+\frac{1}{2N_Q}\sum_{v\in Q}\frac{N_{v,Q}(k)}{N_Q}.\end{aligned} \tag{7}$$

Similar to the case of two segments, SD can be decomposed into more than two segments.

Definition 3: **Segmented Symmetric Degree.** The segmented symmetric degree is defined as the distributional discrepancy between the ith segment of P to Q that

$$\begin{aligned}ssd_{P_i,Q}(k) =& \frac{1}{N_P N_Q}\{\sum_{u\in P_i}\left(-\frac{N_Q}{N_P}-1\right)N_{u,Q}(k)-\\ &\sum_{v\in Q}N_{v,Pi}(k)\}+\frac{kN_{P_i}}{N_P}+\frac{1}{i_{max}N_Q}\sum_{v\in Q}\frac{N_{v,Q}(k)}{N_Q}.\end{aligned} \tag{8}$$

$ssd_{P_i,Q}$ measures the distributional discrepancy from Q to each subset of P. A smaller $ssd_{P_i,Q}$ indicates a better presentation of Q by P_i.

C. Drift-gradient and how to sequentially update it

If we wait for an entire batch of data to detect drift and then update the predictor (if drift is detected), which causes adaptation delay. To overcome the adaptation delay, we propose a sequentially updated statistic, called Drift-gradient.

Drift-gradient is to quantify the increase of SSD. We have discussed in the previous subsection that a smaller $ssd_{P_i,Q}$ indicates a better presentation of Q by P_i. Therefore, a lower value of drift-gradient on ith segment represents that the distribution of Q is *closer* to the distribution of the ith segment. In this section, we will explain how to update the drift-gradient with computing the values of SSD.

We notice that SSD for different segments have common items. Therefore, we simplified SSD from two aspects: 1. we consider P is evenly segmented that $N_{P_i} = N_0$; 2. The common items in $ssd_{P_i,Q}$ is excluded to computing drift-gradient as it does not affect the result.

Given P and its segments with $N_{P_i} = N_0$, the common items for all i in $ssd_{P_i,Q}(k)$ is (9), leaving the discrepancy part of $ssd_{P_i,Q}$ as is in (10). Clearly, $ssd_{P_i,Q}(k) = C(k) + \delta_{P_i,Q}(k)/N_P N_Q$

$$C(k) = \frac{kN_0}{N_P}+\frac{1}{2N_Q}\sum_{v\in Q}\frac{N_{v,Q}(k)}{N_Q} \tag{9}$$

$$\delta_{P_i,Q}(k) = -\sum_{v\in Q}N_{v,Pi}(k) - (1+\frac{N_Q}{N_P})\sum_{u\in P_i}N_{u,Q}(k) \tag{10}$$

Compared to Definition 3, (10) is more concise. The task of quantifying the increase of SSD can be equivalently transferred to the task of quantifying the increase of δ.

Definition 4: **Drift-Gradient.** Given the simplified SSD at a specific time point denoted by δ, drift-gradient is defined as

$$\nabla\delta_i(k,t) = \delta_{P_i,Q_t}(k) - \delta_{P_i,Q_{t-1}}(k) \tag{11}$$

Given P obtained from the past, contains N_P data instances and Q the upcoming batch (batch size: N_Q) of data instances arrives instance by instance, the key to online compute $\nabla\delta$ is to determine the relationship between $\delta_{P_i,Q_t}(k)$ and $\delta_{P_i,Q_{t-1}}(k)$, where Q_t represents the state of Q at time t. Clearly, $Q_t = Q_{t-1} \cup v_t$ where v_t is the t-th data instance. It should be noticed that drift-gradient is not a mathematical gradient. δ is discrete because of $t \in \mathbb{Z}^+$. Therefore, $\nabla\delta$ is not computed by the derivative of a function of a real value. It is called drift-gradient because it denotes the rate of change of δ, which is similar to a gradient denoting the rate of change of a function.

Next, we will discuss how to compute drift-gradient when every new instance arrives. As shown in (11) and (10), drift-gradient is defined as the change of $\delta_{P_i,Q_t}(k)$, and $\delta_{P_i,Q_t}(k)$ consists mainly of $\sum_{v\in Q} N_{v,Pi}(k)$ and $\sum_{u\in P_i} N_{u,Q}(k)$. If we know how $\sum_{v\in Q} N_{v,Pi}(k)$ and $\sum_{u\in P_i} N_{u,Q}(k)$ changes from their previous values, we **do not need to compute** the value of $\delta_{P_i,Q_t}(k)$ and $\delta_{P_i,Q_{t-1}}(k)$ to obtain drift-gradient. Therefore, the sequential update of $\nabla\delta$ contains of two parts when we substitute (10) into (11): 1) the sequential updates of $\sum_{v\in Q} N_{v,Pi}(k)$ denoted by $\nabla\delta_Q$; 2) the sequential updates of $\sum_{u\in P_i} N_{u,Q}(k)$ denoted by $\nabla\delta_{P_i}$. Next, these two parts will be discussed separately.

- *Sequentially update $\sum_{v\in Q} N_{v,Pi}(k)$:*

Given $\sum_{v\in Q_{t-1}} N_{v,Pi}(k)$ represents the number of k-nearest neighbors of current data instances arrived in Q at time point $t-1$, $\sum_{v\in Q_t} N_{v,Pi}(k)$ is computed as

$$\sum_{v\in Q_t} N_{v,Pi}(k) = \sum_{v\in Q_{t-1}} N_{v,Pi}(k) + N_{v_t,Pi}(k). \tag{12}$$

Therefore,

$$\nabla\delta_Q = N_{v_t,Pi}(k). \tag{13}$$

- *Sequentially update $\sum_{u\in P_i} N_{u,Q}(k)$:*

$\sum_{u\in P_i} N_{u,Q}(k)$ can not be directly iterated as the case in $\sum_{v\in Q} N_{v,Pi}(k)$. In order to implement the update, k-nearest neighbor distance matrix is defined as Definition 5.

Definition 5: k**-nearest neighbor distance matrix**. Given a $m \times n$ dimension matrix $A = (\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n)$, its k-nearest neighbor distance matrix is defined as

$$D(k) = \begin{bmatrix} d^1_{(1)} & d^1_{(2)} & \cdots & d^1_{(k)} \\ d^2_{(1)} & d^2_{(2)} & \cdots & d^2_{(k)} \\ \vdots & & \ddots & \vdots \\ d^n_{(1)} & d^n_{(2)} & \cdots & d^n_{(k)} \end{bmatrix}, \tag{14}$$

where $d^j_{(k)}, (j = 1, \cdots, n)$ is the kth order of distance from $\boldsymbol{a}_j$ to A. The first column of $D(k)$ is $\boldsymbol{0}$ because $d^j_{(1)}$ is always the distance from $\boldsymbol{a}_j$ to itself. For example, $A = (\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3) = \begin{bmatrix} 1 & 0 & 5 \\ 3 & 2 & 1 \end{bmatrix}$, its distance matrix is $\begin{bmatrix} 0 & \sqrt{2} & \sqrt{20} \\ \sqrt{2} & 0 & \sqrt{26} \\ \sqrt{20} & \sqrt{26} & 0 \end{bmatrix}$, and corresponding 3-nearest neighbor distance matrix $D(3) = \begin{bmatrix} 0 & \sqrt{2} & \sqrt{20} \\ 0 & \sqrt{2} & \sqrt{26} \\ 0 & \sqrt{20} & \sqrt{26} \end{bmatrix}$. Similarly, $D(2) = \begin{bmatrix} 0 & \sqrt{2} \\ 0 & \sqrt{2} \\ 0 & \sqrt{20} \end{bmatrix}$.

Given $\boldsymbol{a}_j = (\boldsymbol{X}_j, y_j)^T$ and $P_i = (\boldsymbol{a}_1, \cdots, \boldsymbol{a}_{N_{P_i}})$ where $j \in (1, \cdots, N_{P_i})$, its k-nearest neighbor matrix $D_i(k)$ can be computed by (14), denoted by $D_i(k) = (\boldsymbol{d^1}, \boldsymbol{d^2}, \cdots, \boldsymbol{d^{N_{P_i}}})^T$. Given $D_i(k)$ and $Q = (v_{t_0+1}, v_{t_0+2}, \cdots, v_{t_0+N_Q})$ arriving by instance after t_0, updating $\sum_{u\in P_i} N_{u,Q}(k)$ is to compare $D_i(k)$ and the distance from v_t to P_i —$d(v_t, P_i)$, as well as $d(v_t, P_i)$ and its previous values $d(v_{t-1}, P_i), d(v_{t-2}, P_i), \cdots$. Next, the updating process for an arbitrary $u \in P_i$ will be discussed.

Given P_i and its k-nearest neighbor distance matrix $D_i(k)$, for an arbitrary $u \in P_i$, the ordered k-nearest neighbor distance of u is one row of $D_i(k)$, denoted as $\boldsymbol{d}^u$. $d_{v_t,u}$ represents the distance between u and the newly arrived instance v_t from Q. The one-step (one-instance) updating process of $N_{u,Q_t}(k)$ is in **Algorithm** 1 where the updated variable are denoted with t. K_d counts how many items in $\boldsymbol{d}^u$ are larger than $d_{v_t,u}$. $K_d \le 1$, which means the newly arrived v_t is no nearer to u than u's current nearest neighbors, and therefore $N_{u,Q_t}(k) = N_{u,Q_{t-1}}(k)$. Once $K_d > 1$, it also needs to consider whether a previous v is excluded from u's k-nearest neighbors, as v_t becomes u's k nearest neighbor. To implement that, the variable o_t is introduced to record the order of $d_{v_t,u}$ and the furthest neighbor in $\boldsymbol{d}^u$ will be deleted from $\boldsymbol{d}^u$ (implemented in line 8). By this design, when the length of current $\boldsymbol{d}^u$ is less than $\max \boldsymbol{O}_{Q_t}$, this denotes that the previous $d_{v,u}$ are still less than the furthest neighbor from P_i. Therefore, $N_{u,Q_t}(k) = N_{u,Q_{t-1}}(k) + 1$. Otherwise, $N_{u,Q_t}(k)$ will not update, because although v_t becomes one of u's k-nearest neighbors, a previous v is excluded from the neighbors at the same time. An intuitive perception of this design is that δ_Q is only updated when the threshold for v becoming u's k-nearest neighbors is leveled up.

Given $\sum_{u\in P_i} N_{u,Q_{t-1}}(k)$, $\sum_{u\in P_i} N_{u,Q_t}(k)$ can be updated by the sum of $N_{u,Q_t}(k)$ and $\nabla\delta_{P_i} = \sum_{u\in P_i} N_{u,Q_t}(k) - N_{u,Q_{t-1}}(k)$. When a new instance from Q arrives, the drift-gradient on each segment in P is computed as $\nabla\delta_i = \nabla\delta_Q + \nabla\delta_{P_i}$.

To validate the effectiveness of the proposed drift-gradient, we conduct experiments of whether the drift-gradient can correct identify the most appropriate segment in the training set. We generate 50-dimensional instances from three different uniform distributions denoted by P1, P2 and P3. P1 is generated by numpy.random.seed(1) in python, P2 is generated with the mean 0.4 larger than P1, and P3 is generated with the mean 0.4 larger than P2. The training set consists of three segments, which are shown in Table I. Then we generate 200 instances from P3, and 200 instances from P2 as the testing set, and they arrive one instance at a time. In Experiment1, P3 arrives before P2, while in Experiment2 P2 arrives before P3. In Table II, the Type I error of P1 is how many true P1 instances are correctly labelled by P1 on average. As there

Algorithm 1: One-step update of $N_{u,Q_t}(k)$

```
Input : $\boldsymbol{d}^u$, $d_{v_t,u}$, $\boldsymbol{O}_{Q_{t-1}}$, $N_{u,Q_{t-1}}(k)$.
Output: $N_{u,Q_{t-1}}(k)$
Initialization: $\boldsymbol{O}_{Q_0} \leftarrow [\ ]$
1  Compute $K_d = |\boldsymbol{d}^u > d_{v_t,u}|$;
2  if $K_d > 1$ then
3      % the new instance is nearer to u than u's current neighbors
4      Compute $o_t = |\boldsymbol{d}^u| + 1 - K_d$;
5      Compute $\boldsymbol{O}_{Q_t} = \left[\boldsymbol{O}_{Q_{t-1}}; o_t\right]$
6      if $|\boldsymbol{d}^u| \le \max \boldsymbol{O}_{Q_t}$ then
7          Update $N_{u,Q_t}(k)$=$N_{u,Q_{t-1}}(k) + 1$;
8          Delete $\boldsymbol{d}^u$[end];
9      else
10         % a previous v is excluded from u's neighbors
11         Update $N_{u,Q_t}(k) = N_{u,Q_{t-1}}(k)$;
12     end
13 else
14     % the new instance is no nearer to u than u's current neighbors
15     Update $N_{u,Q_t}(k) = N_{u,Q_{t-1}}(k)$;
16 end
17 return $N_{u,Q_t}(k)$, $\boldsymbol{O}_{Q_t}$
```

are no instances from P1 in the new instances, this value is 0. The Type II error of P1 concerns the number of true P2 and P3 instances wrongly labelled by P1 on average. As there are 10 P2 instances wrongly labelled by P1 in Experiment1 in the new instances, this value is 10/400. Similar computation is conducted on P2 and P3. The Type I and Type II error shows drift-gradient is able to select the segment that is closer to the new instances.

D. A segment-based adaptation method—SEGA

The previous subsection solved how to sequentially update drift-gradient. Based on this solution, an online drift adaptation method, SEGA is presented in this section.

For a data stream $\{D_t = (\boldsymbol{X}_t, y_t), t = 1, \cdots, \infty\}$ with $\boldsymbol{X}$ its attributes and y the corresponding label, $\{D_t, t \le T_0\}$ is assumed to be already obtained as the historical data. The data instances after T_0 will arrive one by one, by time and $\boldsymbol{X}_t$ is observed before y_t. The online prediction is to first apply the current trained predictor to predict y_t for $t > T_0$ given the value of $\boldsymbol{X}_t$. After the true value of y_t is obtained, this newly arrived D_t will be used to update the current predictor.

This paper uses an update process that combines batch and online adaptation. Before a full batch of new instances arrives, an online adaptation is activated to tune the current predictor, and once a full batch of new instances is obtained, the current predictor will be retrained as is in a batch adaptation. **Algorithm** 2 presents how SEGA update the predictors from $\hat{H}_{t-1}$ to $\hat{H}_t$. The parameter w denotes the length of the segments in the training set. s presents the number of segments that each training set contains. During the experiments, the s is assigned a fixed size and a period of data instances of $s \times w$ will be picked from the historical data as the initial training data. c is an ensemble coefficient to control how many segments are

Algorithm 2: SEGA

```
Input : $D_t$, $\hat{H}_{t-1}(\cdot|\boldsymbol{\Theta})$, $P$, $Q$, $w$, $c$, $s$.
Output: $\hat{H}_t$, $P$, $Q$.                  % $\hat{H}_t$ is used to predict $y_{t+1}$
1  $Q = [Q; D_t]$ ;
2  if $|Q| < w$ then
3      % sequentially update predictors and store new instance in Q
4      for $P_i$ in $P$ do
5          % compute drift gradient for each segment
6          Compute $\nabla\delta_Q$ by (13);
7          for $u$ in $P_i$ do
8              Update $N_{u,Q_t}(k)$ by Algorithm 1;
9              $\nabla\delta_{P_i} \leftarrow \nabla\delta_{P_i} + (N_{u,Q_t}(k) - N_{u,Q_{t-1}}(k))$
10         end
11         $\nabla\delta_i = -\nabla\delta_Q - (1 + w/|P|)\nabla\delta_{P_i}$
12     end
13     $i_c = \underset{i}{\arg\text{sort}}\ \nabla\delta_i[1:c]$; %c segments with minimal drift gradient;
14     $\hat{H}_c(\cdot|\boldsymbol{\Theta}) = \frac{1}{c}\sum_{i\in i_c}\hat{H}_i(\cdot|\boldsymbol{\Theta})$;      % combined by average;
15     $\hat{H}_t(\cdot|\boldsymbol{\Theta}) \leftarrow \hat{H}_c(\cdot|\boldsymbol{\Theta})$
16 else
17     % a batch of new data has been stored, initializing the buffer of
         the new concept Q
18     Segment $P$ into $P_1, \cdots, P_s$;
19     $P \leftarrow [Q, P_2, \cdots, P_s]$;       % the training set is updated;
20     $\hat{H}_t(\cdot|\boldsymbol{\Theta}) \leftarrow \hat{H}_1(\cdot|\boldsymbol{\Theta})$ ;
21     $Q = [\ ]$;    % initialize Q
22 end
```

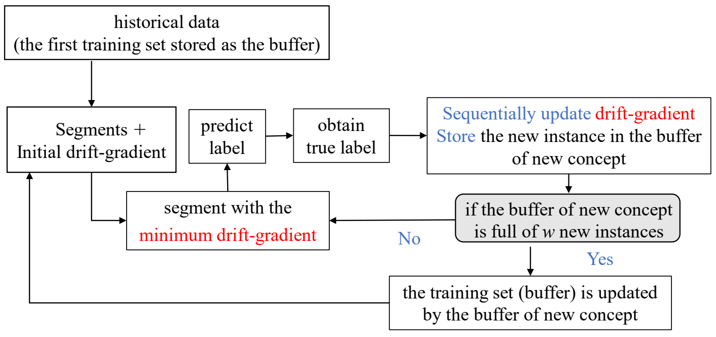

historical data
(the first training set stored as the buffer)
Segments + Initial drift-gradient
predict label
obtain true label
Sequentially update drift-gradient
Store the new instance in the buffer of new concept
segment with the minimum drift-gradient
if the buffer of new concept is full of w new instances
No
Yes
the training set (buffer) is updated by the buffer of new concept

Fig. 3. Flowchart of SEGA.

picked. The row 7-10 updates δ_P by $\nabla\delta_P$ can be skipped for a faster computation with slightly lower accuracy.

The flowchart of SEGA is shown as Fig. 3. The training set will be separated into disjointed segments and the drift-gradient is computed to dynamically select the best segments for prediction. If the whole training set is used to train the predictor and retrain every buffer of new concept or every instance, it is a sliding window adaptation. In the experiment section, the sliding window adaptation will be compared as a baseline method, to show the necessity of segments and the effectiveness of drift-gradient for handling the concept drift problem.

TABLE I
VALIDATION OF THE EFFECTIVENESS OF DRIFT-GRADIENT

Experiment1	segment1	segment2	segment3	new instance arrives one by one					
numbers of instances	400	400	400		200			200	
true distribution	P1	P2	P3		**P3**			**P2**	
lablled by drift-gradient	-	-	-	P1	P2	P3	P1	P2	P3
numbers of instances	-	-	-	0	2	197	10	173	17
Experiment2									
true distribution	P1	P2	P3		**P2**			**P3**	
lablled by drift-gradient	-	-	-	P1	P2	P3	P1	P2	P3
numbers of instances	-	-	-	3	183	14	0	11	189

TABLE II
TYPE I AND TYPE II ERROR OF DRIFT-GRADIENT UNDER THREE DISTRIBUTIONS

	Experiment1		**Experiment2**		**Average**	
	Type I	Type II	Type I	Type II	Type I	Type II
P1	0	10/400	0	3/400	0	0.01625
P2	1-173/200	2/200	1-183/200	11/200	0.11	0.0325
P3	1-197/200	17/200	1-189/200	14/200	0.035	0.0775

IV. EXPERIMENTAL EVALUATIONS

In this section, SEGA is compared to 15 benchmarks on 12 synthetic data and 14 real-world data streams. The **experimental results and analysis** are given in section IV-A and IV-B. As the predictors used for SEGA in tested data streams are different for regression and classification tasks, the **experimental configuration** will be introduced and specified at the beginning of each subsection. **Friedman test** of comparison between SEGA and other baseline methods are conducted in section IV-C. The **parameter analysis and computation complexity** are presented in section IV-D.

Four kinds of data streams are involved: synthetic regression data, synthetic classification data, real-world regression data and real-world classification data. To our best knowledge, this is the first time that the concept drift problem has been comprehensively tested on both continuous and discrete label variables. It is not known if a data stream contains drift. The types of drift that exists in a real-world data stream is also unknown, Therefore, drift is manually added into the synthetic data streams. Experiments on the synthetic data streams are to validate that SEGA can solve concept drift problems. In the experiments on synthetic data, all types of drift have been evolved. Details of drift types will be introduced in each subsection of experiments.

Data streams are supposed to be infinite but to validate the algorithm and present its effectiveness, it is essential to obtain a finite period of data streams. A common way to evaluate the algorithm effectiveness on data streams is prequential evaluation, where each data instance is first used to test the predictor, and then to train the predictor. In this paper, a fixed length of historical data instances are available before conducting the experiments. Future data instances are available one by one during the experimental procedure. In this paper, we consider prediction accuracy as the validation criterion including *mean absolute error* (15) in regression tasks and *accuracy* in classification tasks (16).

$$\mathrm{MAE} = \frac{1}{(T-t_0)} \sum_{t=t_0}^{T} |\hat{y}_t - y_t| \,, \tag{15}$$

$$\mathrm{Acc} = \frac{TP+TN}{TP+TN+FP+FN}, \tag{16}$$

In the following two subsections, the experimental results of SEGA on synthetic and real-world data streams will be displayed and analyzed. In both subsections, the experiments contain regression cases and classification cases.

A. Evaluation on Synthetic Data

In this section, SEGA will be evaluated on six synthetic data of regression tasks and ten synthetic data of classification tasks. The aim of the experiments on synthetic data is to validate that SEGA is effective to handle drift problem. Therefore, SEGA will be compared to its corresponding non-adaptation edition on data streams containing different types of drift, including no drift. As discussed in section III-D, SEGA will also be compared to its sliding window edition to validate the effectiveness of segment.

Data Description. The synthetic data of regression data is the same as the synthetic data in [22], which includes one original data stream that does not contain drift, one data stream containing virtual drift and four data streams with real drift. Since the drifted data is generated based on the non-drifted data, it is clear to understand the extent that SEGA solves the problem of concept drift in a data stream.

For the classification task, ten widely used synthetic data is introduced to validate the effectiveness of SEGA on the drift problem in the classification task. All the synthetic data are available from [37].

Configuration. In the regression tasks, the predictor in SEGA is a linear predictor with $L2$-norm (ridge regression) where $\alpha = 0.01$, each segment is of 100 instances, the training set contains 10 segments (namely, the length of training set is 1000), and the ensemble coefficient $c = 6$.

In the classification tasks, the predictor in SEGA is a weighted k nearest neighbor (kNN) classifier with $k = 5$, the length of each segment is 200, the training set contains 10 segments, and the ensemble coefficient $c = 1$.

Analysis of Results on Synthetic Regression Tasks. We compare our method to its non-adaptation version and sliding window version. The results are presented in Table III. In addition, we also present a comparison result of our method

TABLE III
VALIDATION OF REGRESSION TASKS ON SYNTHETIC DATA (MAE AS THE EVALUATION CRITERION).

Data Streams	**Data Description**		**Tested Models**			**Model Effectiveness**	
	instances	Drift Type	Linear	SlidWin	**SEGA-Linear**	SEGA-Ideal	Effectiveness
Non-Drift	2000	no drift	0.800	0.811	0.810	-	-
Virt-Drift	2000	sudden virtual drift	0.780	0.798	0.793	-	-
Sudd-Drift	2000	sudden real drift	13.540	2.728	2.717	2.592	95.2%
Incr-Drift	2000	incremental real drift	10.200	2.315	2.307	2.220	96.1%
Rec-Drift-Grad	12000	sudden and gradual real drift	8.800	9.118	1.265	1.047	79.2%
Rec-Drift-Mix	12000	sudden, incremental and reoccurring real drift	8.170	2.486	1.680	1.584	93.9%

TABLE IV
VALIDATION ON SYNTHETIC DATA OF CLASSIFICATION TASKS (ACC AS THE EVALUATION CRITERION).

Data Streams	**Data Description**	**Tested Predictors**		
	instances	kNN	SlidWin	**SEGA**$_{kNN}$
SEAs(s1)	10,000	80.95%	81.85%	84.41%
SEAg(g1)	10,000	81.00%	81.79%	84.60%
HYPER(i& r)	10,000	77.38%	78.34%	78.67%
AGRs(s1)	10,000	87.95%	61.66%	61.99%
AGRg(g1)	10,000	87.95%	61.59%	62.49%
RTG	10,000	73.91%	73.85%	65.69%

(s): sudden drift; (i):incremental drift; (g): gradual drift; (r): reoccurring concepts
(0): virtual drift; (1): real drift

with other existing methods in Table IX just for a reference. In Table III, the column, Data Description, lists the length of each data and the type of drift it contains. Prediction accuracy is shown in the Tested Models column. Linear column denotes a ridge regression predictor that's trained on the first training set and is used for all the testing data without retrain. The SlidWin column presents the prediction results of a sliding window edition. In the sliding window edition, the training set will be updated by including the newly arrived instance and discarding the oldest instance. For each newly arrived instance, a new predictor will be trained on the current training set, to predict the label for the next instance. SlidWin can partly solve the problem of concept drift, as it obtain better prediction accuracy than Linear on some data streams with real drift. Our method, displayed in the SEGA-Linear column, uses a drift-gradient to choose the best segment for prediction to adapt to new concept. Model Effectiveness column evaluates SEGA to determine how much SEGA differs from an ideal edition, where the drift-gradient correctly designates the best segment for each tested instance. The MAE in the **SEGA-Ideal** column is computed as follows:

- Predict the label by the trained predictor on each segment. As the segment number is 10, there are 10 predicted value for each test instance.
- Given the **true value of the label**, choose the best result from 10 predictions with the minimum absolute error for the prediction of this instance.
- Compute MAE. This is the minimum MAE that SEGA could obtain, because it is computed with the true label.

The Effectiveness column is computed by MAE of SEGA-Linear and SEGA-Ideal that

$$\text{Effectiveness} = \frac{MAE_{SEGA-Linear}}{MAE_{SEGA-Ideal}}. \tag{17}$$

SEGA-Linear is more applicable on the data with a larger value of the effectiveness.

The MAE results in Table III shows that:

1) Linear, SlidWin and SEGA-Linear performs the same when data contains no drift or virtual drift. The synthetic data in Table III are generated by a linear function with a random error, and a linear predictor is used during the experiments. Therefore, the experimental results are less affected by the cause of an inappropriate predictor. If the data is generated by a very complex function, and it has to use a powerful predictor trained on a large size of instances to accurately present the true hypothesis, the effectiveness of solving concept drift will highly be impaired by the poor performance of the predictor trained on a limited size of instances. In the Non-Drift data stream, it can be seen that the prediction accuracy of Linear, SlidWin and SEGA-Linear are all close to 0.800, which shows that for this data stream, if no drift occurs, there is no difference between learning a predictor on the full training set and on one of its segments. This is also one of the required applicable conditions when choosing the appropriate predictor in SEGA. It has been discussed in [22] that virtual drift has little influence on the prediction results if the change of $p(\boldsymbol{X})$ is independent to $p(y|\boldsymbol{X})$. Here, the same results show — Linear, SlidWin and SEGA obtain almost the same accuracy on the Virt-Drift stream.

2) Simple retraining is not suitable for all types of drift. By comparing the Linear, SlidWin and SEGA-Linear columns in Table III. SlidWin is to simply retrain the current predictor when every new instance arrives, and does not include any design on how to use drift information to help adaptation. According to the value of MAE of SlidWin, it performs well on most data streams and can partly handle the concept drift problem under some conditions. For example, the MAEs of SlidWin are much smaller than the corresponding MAEs of Linear on Sudd-Drift, Incr-Drift and Rec-Drift-Mix. Specifically, SlidWin is as good as SEGA on Sudd-Drift and Incr-Drift streams. However, SlidWin can not fully solve the reoccurred concept problem. SlidWin performs worse than SEGA on two streams containing reoccurred drift, and even worse than the Linear model on the Rec-Drift-Grad stream.

3) Retraining by segment is an effective way to solve concept drift problem. Our proposed SEGA method can outperform other method when any type of drift or mixture of occurs.

4) The proposed SEGA method can accurately identify the best segment in the training set by drift-gradient. Comparing the column of SEGA-Linear and SEGA-Ideal, it can be seen

TABLE V
VALIDATION ON REAL-WORLD DATA OF REGRESSION TASKS (MAE AS THE EVALUATION CRITERION).

MAE(rank)	ORTO	FIMT-DD	metaAMR	AMR	Per	FUZZ-CARE	SlidWin$_{ridge}$	**SEGA**$_{ridge}$
CCPP	4.53E+02(8)	3.57E+00(3)	3.32E+00(1)	3.42E+00(2)	3.65E+00(4)	5.59E+00(7)	3.67E+00(6)	3.67E+00(5)
Sensor3	6.62E-02(8)	7.14E-03(3)	1.58E-02(6)	7.69E-03(4)	6.85E-03(2)	1.55E-02(5)	3.58E-02(7)	6.13E-03(1)
Sensor8	1.69E-01(8)	9.68E-03(5)	7.26E-03(4)	6.57E-03(2)	5.95E-03(1)	1.72E-02(6)	7.57E-02(7)	7.14E-03(3)
Sensor20	9.60E-01(8)	7.95E-01(6)	1.14E-02(4)	8.19E-03(3)	7.92E-01(5)	7.90E-03(2)	8.08E-01(7)	7.35E-03(1)
Sensor46	4.00E-01(7)	1.57E-01(4)	1.74E-01(5)	2.02E-01(6)	1.56E-01(3)	5.25E-02(2)	5.10E-01(8)	5.87E-03(1)
SMEAR	3.39E+01(7)	2.34E+01(5)	1.98E+01(4)	1.44E+01(2)	3.75E+01(8)	1.04E+01(1)	2.94E+01(6)	1.73E+01(3)
Solar	2.25E+02(8)	1.14E+02(5)	9.39E+01(2)	9.52E+01(3)	1.30E+02(6)	8.66E+01(1)	2.17E+02(7)	1.09E+02(4)
AvgRank(no CCPP)	7.67	4.67	4.17	3.33	4.17	2.83	7.00	**2.17**
AvgRank	7.71	4.43	3.71	3.14	4.14	3.43	6.86	**2.57**

that when real drift occurs, the prediction accuracy of SEGA-Linear is very close to the accuracy of SEGA-Ideal. As SEGA-Ideal predicts the label based on the true value of that label, it is considered to be the optimal prediction if using one segment of training set to predict labels. According to the Model Effectiveness result, our proposed SEGA can obtain 95.2% prediction capability of a prediction where the true label is known on Sudd-Drift, and a prediction capability of 96.1%, 79.2% and 93.9% on Incr-Drift, Rec-Drift-Grad and Rec-Drift-Mix respectively. This demonstrates the effectiveness of the drift-gradient in SEGA.

Analysis of Results on Synthetic Classification Tasks. Similar to the regression case, we compare SEGA to its non-adaptation version and sliding window version in classification case and the results are presented in Table IV. The comparison of SEGA to other existing methods is presented in Table X for a reference. In Table IV, these synthetic data contain various types of drift. In classification tasks, kNN is applied as the basic predictor. The model effectiveness is not tested in classification tasks, since the prediction accuracy of SEGA-Ideal is more likely to be 1. This accuracy of 1 is largely due to the randomness, rather than the 100% accurate predictor. Therefore, it is not able to reflect the true capability of SEGA-Ideal. The prediction results of classification tasks show that:

1) Retrain by segments is better than Retrain by the whole training set. This is concluded because according to Table IV, SEGA is no worse than SlidWin on all synthetic classification data streams with drift (noticing that RTG does not contain drift).

2) SEGA may not be suitable in cases where low-frequent and small drift exists. In some data streams such as AGRs, although it contains drift, the accuracy of SlidWin is much worse than that of kNN, which denotes that non-adaptation is a better choice for this drifted data. This is because the drift in AGR data occurs at a low frequency and the new pattern has small difference from the previous one.

B. Evaluation on Real-world Data Streams

In this section, SEGA will be evaluated on eight real-world data streams of regression tasks and seven real-world data streams of classification tasks. In the previous subsection, SEGA has been validated to solve concept drift effectively. The aim of this subsection is to compare it to some state-of-art regression and classification methods, which are specially designed for solving concept drift problems.

Data Description. The seven real-world regression data streams are: CCPP containing 9568 instances with four attributes; Sensor 3, 8, 20 and 46 containing 46,633, 15,808, 28,832 and 52,988 respectively with three attributes; SMEAR containing 140,576 instances with 43 attributes and Solar containing 32,686 instances with five attributes. Detailed information could be found in [22] with the download like in [38]. Among them, CCPP has been validated not to contain the concept drift problem [22], [39].

The seven real-world classification data streams are: Elec (normalized) containing 45,312 instances with eight attributes; Weather containing 18,159 instances with eight attributes; Spam containing 9,324 instances with 39,916 attributes; Airline containing 539,383 instances with eight attributes; Covertype (normalized) containing 45,312 instances with 9 attributes; Usenet1 containing 1,500 instances with 99 attributes; and and Usenet2 containing 1,500 instances with 99 attributes. More details of these data streams can be found in [16], [40].

Configuration. In the regression tasks, the predictor in SEGA is ridge regression with $\alpha = 1e-04$ for CCPP, Sensor3, Sensor20, and Sensor46; $\alpha = 1e-03$ for Sensor8; $\alpha = 1e-02$ for SMEAR; and $\alpha = 1e-01$ Solar. The value of α is determined by parameter analysis, which will be provided in the next section. The length of each segment is 400 for SMEAR and 200 for the other data streams, because SMEAR has more attributes and a size of 200 is not large enough for training a predictor in this case. The training set contains 10 segments, and the ensemble coefficient $c = 2$. The k for searching nearing neighbors is half of the length of the segment. More details of selecting the parameters can be found in Section IV-D.

In the classification tasks, the predictor in SEGA is a weighted k-nearest neighbor classifier with $k = 5$. The length of each segment is 200 and each training set contains 10 segments for the data streams, except for Usenet1 and Usenet2, because Usenet1 and Usenet2 only have 1500 instances in total. For Usenet1 and Usenet2, the training set contains 7 segments. The ensemble coefficient $c = 6$. The k for searching nearing neighbors is half of the length of the segment. More details of selecting the parameters can be found in Section IV-D.

Analysis of Results on Real-world Regression Tasks. In the experiments of real-world regression data streams, the effectiveness of SEGA is presented by comparing it to six benchmark drift adaptation methods aiming to handle concept

TABLE VI
VALIDATION ON REAL-WORLD DATA OF CLASSIFICATION TASKS (ACC AS THE EVALUATION CRITERION).

	ADWIN-ARF	NN-DVI	LevBag$_{kNN}$	SAM$_{kNN}$	OnlineAUE	IBLStream$_{kNN}$	Learn++NSE$_{kNN}$	SlidWin$_{kNN}$	SEGA$_{kNN}$
Elec	88.17(1)	86.67(3)	81.91(6)	82.78(5)	87.74(2)	77.05(7)	70.52(8)	62.81(9)	83.46(4)
Weather	78.74(2)	74.75(8)	76.19(5)	77.73(3)	75.24(7)	75.69(6)	68.27(9)	77.11(4)	79.29(1)
Spam	95.60(2)	94.65(3)	93.22(5)	95.79(1)	84.29(7)	92.78(6)	70.56(8)	66.90(9)	94.57(4)
Airline	65.24(2)	65.20(3)	65.03(4)	60.35(8)	67.51(1)	63.74(5)	62.35(6)	53.90(9)	61.56(7)
Covertype	92.11(5)	94.04(2)	94.00(3)	91.71(6)	90.01(7)	92.26(4)	64.03(9)	71.88(8)	94.72(1)
Usenet1	68.40(2)	61.40(6)	58.93(7)	65.67(4)	63.47(5)	56.00(8)	48.53(9)	67.00(3)	80.00(1)
Usenet2	71.93(1)	71.40(2)	67.33(8)	71.00(3)	68.87(6)	67.67(7)	66.67(9)	70.00(5)	71.00(3)
AveRank	2.14	3.86	5.43	4.29	5.00	6.14	8.29	6.71	**3.00**

drift problems in a regression task. The benchmarks are ORTO [41]; FIMT-DD [42]; AMR and metaAMR [43]; Perceptron [44]; and FUZZ-CARE [22]. ORTO and FIMT-DD are tree model based methods, which use linear regression models and the stochastic gradient descent method in the leaves of the tree. ORTO and FIMT-DD detect the drift by Page-Hinckley (PH) test and use the detection result to adjust the tree structure. AMR and metaAMR are rule models and ensemble rules. For each rule model, a linear regression model is trained by an incremental gradient descent method. Perceptron is a Hoeffding perceptron tree model which replaces the naive Bayes with perceptron predictor. FUZZ-CARE is implemented by the code in [38] and all the other benchmarks are implemented by MOA [40] with their default parameters. SlidWin uses ridge regression as the basic predictor.

The results of SEGA on real-world regression tasks are listed in Table V with the following concludes:

1) Using the whole training set is not the best strategy for drift adaptation. According to the average rank results, it can be seen that SlidWin is the second worst drift adaptation method among these benchmarks. This phenomenon denotes that using the whole training set during the adaptation procedure is not a wise choice for the real-world data streams, because the drift situation is very complex in the real world.

2) Our proposed SEGA is able to handle different drift cases in the real-world data. In [22], the authors have discussed that among the data streams in Table V, Sensor20, Sensor46, SMEAR and Solar, are supposed to have significant reoccurring drift according to their experiments. As FUZZ-CARE is specially designed for reoccurring drift, it obtain better performance on these five data streams. Our proposed SEGA does not aim at solving a special type drift but is designed to be suitable for the occurrence of all types of drift. The highest average rank of SEGA compared to the other benchmarks validates the effectiveness of SEGA to solve concept drift problems in a data stream.

Analysis of Results on Real-world Classification Tasks. For real-world classification data streams, the benchmarks here are all designed to specially solve the concept drift problem in the data stream of classification tasks, including ADWIN-ARF which uses ADWIN to detect drift and use an adaptive random forests for classification [45]; NN-DVI, which detects drift via a density based distance and adapts to a new concept via competence model [16]; LevBag is ensemble method using an improved online bagging method to adapt to the changing data [46]; SAM$_{kNN}$ ensembles two sliding window, with different window sizes on the kNN classifier [47]; OnlineAUE [26] ensembles the classifier trained from blocks of training data which is similar to the segments in SEGA. OnlineAUE learns the weights of each block but SEGA uses the drift-gradient to select the best segment; IBLStream uses the instance-based model to adapt to the new concept by autonomously optimizing the size of the case base [48]; Learn++NSE is an ensemble method which use the tie-adjusted accuracy to determine weights [49]. NN-DVI is implemented with window size of 200 and default parameters, and the other benchmarks are implemented with default parameters by MOA. SlidWin uses the same basic classifier with SEGA, which is the kNN classifier.

The experimental results of SEGA on real-world classification tasks are listed in Table VI.

1) SEGA is also suitable to handle concept drift problems in classification tasks. Compared to the benchmarks which are specially designed to solve the concept drift problem in classification tasks, SEGA has the second highest average rank which validates the power of SEGA to solve the drift problem. In addition, the accuracy of SEGA can be further improved if appropriate predictors are chosen. For example, if decision tree is applied, the accuracy of SEGA on Elec will be 88.48%. We encourage users to try different predictor and ensemble parameters for getting better results of SEGA when they apply SEGA into a specific case.

2) Combining the prediction results of selective segments by drift-gradient is an effective way to implement ensemble. Most of the compared benchmarks are ensemble methods. Some benchmarks ensemble different classifiers while others ensemble the results computed on different data chunks which is similar to the SEGA method. OnlineAUE ensembles the results of data chunks. Therefore, OnlineAUE can particularly be compared to our SEGA method. According to the result table, SEGA obtain an average rank of 3.00, while OnlineAUE obtain 5.00. It can be seen that SEGA is much better than OnlineAUE on the average performance of the tested data streams. In addition, there is no need to train the ensemble weight in SEGA, which means SEGA is much quicker to implement ensemble. Therefore, it is not always recommended to ensemble all the available information. Selecting the most useful information from the training set is an more important aspect for drift adaptation.

- Experiment summary

In this paper, the proposed SEGA has been validated and compared from two aspects: experiments on synthetic data or

TABLE VII

FRIEDMAN TEST AND ITS POST-HOC TEST OF ALL THE METHODS OVER REAL-WORLD REGRESSION DATA STREAMS (NO CCPP), WHERE "FRIEDMAN TEST" IS THE RESULT FOR FRIEDMAN TEST AND "FRIEDMAN - POST-HOC TEST AFTER CONOVER" IS FOR THE PAIRWISE COMPARISON. "+", "*", "**", AND "***" MEANS THIS VALUE IS SIGNIFICANT AT THE LEVEL OF 0.1, 0.05, 0.01 AND 0.001 RESPECTIVELY. "DF" DENOTES THE FREEDOM DEGREE.

Friedman Test		χ_R^2			P-value of χ_R^2		df
		26.11			4.81e-2***		7
Post-hoc test after Conover	ORTO	FIMT-DD	metaAMR	AMR	Per	FUZZ-CARE	SlidWin
$R_i - R_{SEGA}$	33	15	12	7	12	4	29
P-value	8.05e-06***	0.015*	0.0216*	0.1481	0.039*	0.2742	4.94e-05***

TABLE VIII

FRIEDMAN TEST AND ITS POST-HOC TEST OF ALL THE METHODS OVER REAL-WORLD CLASSIFICATION DATA STREAMS, WHERE "FRIEDMAN TEST" IS THE RESULT FOR THE FRIEDMAN TEST AND "FRIEDMAN - POST-HOC TEST AFTER CONOVER" SHOWS THE PAIRWISE COMPARISON. "+", "*", "**", "***" AND "DF" HAVE THE SAME MEANING AS THEY ARE IN TABLE VII

Friedman Test		χ_R^2			P-value of χ_R^2			df
		17.600			0.0244*			8
Post-hoc test after Conover	ADWIN-ARF	NNDVI$_{kNN}$	LevBag	SAM$_{kNN}$	OnlineAUE	IBLStream	Learn++NSE	SlidWin
$R_i - R_{SEGA}$	-6	6	16	9	13	21	36	25
P-value	0.264	0.264	0.048*	0.173	0.088+	0.016*	2.00e-4***	0.0055**

real-world data and experiments on regression or classification tasks. Experiments on the synthetic data denotes that SEGA can improve the prediction accuracy because it can truly solve the concept drift problem in the data. Although the predictors in regression and classification are different, SEGA obtain uniformly good results, which denotes that SEGA is suitable to predict different kinds of data streams. Besides, the advanced performance of SEGA compared to the SlidWin method, demonstrates that it is not always best to use the whole training set to build the model when drift occurs, and the drift-gradient mechanism in SEGA can accurately and effectively select the most appropriate segments in training set for prediction.

C. Statistical Test of Real-world Data streams

In this section, the results of the statistical test will be given to validate the significance of SEGA. The Friedman test and its post-hoc test after Conover are introduced as the testing method where the Friedman test is used to validate whether these drift adaptation methods are significantly different in general. Furthermore, the post-hoc test after Conover is used to validate the significance of pairwise comparison between SEGA and other methods. The statistical test includes tests on MAE of real-world regression data streams and Acc of real-world classification data streams.

Given M the number of tested drift adaptation methods and n the number of data streams, the χ_R^2 statistic in the Friedman test is computed in (18) where R is the rank computed by MAE in regression cases and Acc in classification cases.

$$\chi_R^2 = \frac{12}{nM(M+1)} \sum_{m=1}^{M} R_m^2 - 3n(M+1), \tag{18}$$

If the Friedman test reject the null hypothesis which means these drift adaptation methods are different in general, the post-hoc test will further test whether the difference between SEGA and other methods denoted by $R_i - R_{SEGA}$ is statistically significant [1]. $R_i - R_{SEGA}$ is significant if the following condition satisfies, where α is a preassigned significance level.

$$|R_i - R_j| > t_{1-\frac{\alpha}{2};(n-1)(M-1)} \times \sqrt{\frac{2M(1-\frac{\chi_R^2}{n(M-1)})(\sum_{i=1}^{n}\sum_{m=1}^{M} R_{i,m}^2 - \frac{nM(M+1)^2}{4})}{(M-1)(n-1)}}. \tag{19}$$

The results of statistical test are shown in Table VII and Table VIII. According to the statistical test, we can conclude:

1) The Friedman test results are significant on of both regression and classification cases. The p-value of χ^2 in both cases is small which denotes a significant difference in the prediction accuracy among the tested adaptation methods.

2) In regression case, SEGA$_{ridge}$ is significantly better than most drift adaptation methods. In Table VII, all the p-values of the post-hoc test are significant except for the AMR and FUZZ-CARE column. This means, although SEGA is superior to AMR by a 7 rank difference and to FUZZ-CARE by a 4 rank difference, the difference is insignificant. Similarly, AMR and FUZZ-CARE cannot outperform SEGA, and this insignificance does not affect the effectiveness of SEGA handling the concept drift problem.

3) In the case of classification, SEGA$_{kNN}$ is significantly better than other adaptation methods except for ADWIN-ARF. In Table VIII, the p-values of Friedman test is significant except for the comparison between ADWIn-ARF, NNDVI$_{kNN}$, SAM$_{kNN}$ and SEGA$_{kNN}$. Similarly to the regression case, ADWIn-ARF, NNDVI$_kNN$, SAM$_{kNN}$ are not better than SEGA$_{kNN}$.

D. Parameter Analysis and Computation Complexity

Fig. 4 and Fig. 5 shows how the accuracy will change when the size of the segment w and the number of segments in the

[1]the post-hoc test can test whether the difference between any two of the methods is significant. We only present the post-hoc test result between SEGA and other methods because we do not care whether other methods have significant difference between each other. More details of Friedman test and its post-hoc test can refer to [50].

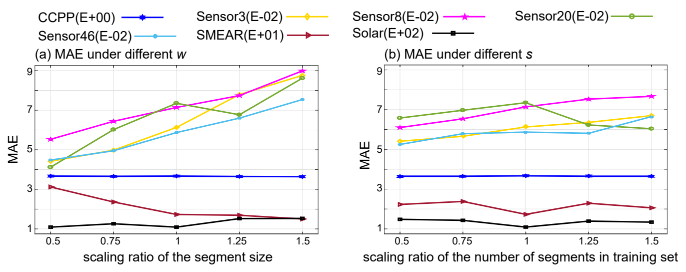


Fig. 4. Parameter analysis for real-world data streams of regression tasks. Different values of the segment size w and the number of segments in the training set s are analyzed. The MAE results of all the data streams are put in the interval $[1, 9]$, and the legend gives the magnitude of each stream. For example, Sensor3(E-02) means the MAE at a specified parameter for this data stream is the value on the y-axis$\times 10^-2$.

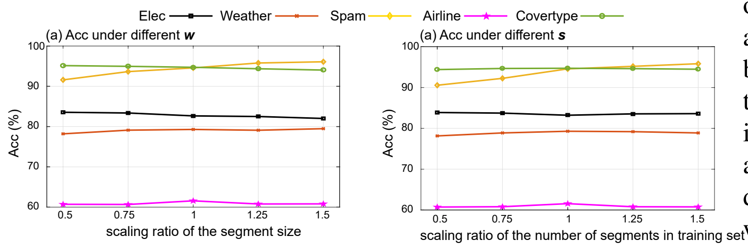


Fig. 5. Parameter analysis for real-world data streams of classification tasks. Usenet1 and Usenet2 are not included because they have few instances.

training set s have different values. We tested different w and s for all the tested real-world data streams by multiplying a scaling ratio with the current used w and s. For example, the accuracy of CCPP listed in Table V, is computed with $w = 200$ and $s = 10$. In the subplot (a) in Fig. 4, the first point in the CCPP line at 0.5 means the MAE is computed with $w = 0.5 \times 200 = 100$ and $s = 10$. For SMEAR, since its accuracy in Table V is computed with $w = 400$, the first point of SMEAR of subplot (a) in Fig. 4, means this MAE is computed with $w = 0.5 \times 400 = 200$ and $s = 10$.

The batch size, which is the segment size w in SEGA, is a critical parameter for drift detection algorithms. For SEGA, if w is too small, the predictor trained on this segment may not be sufficiently trained. If w is too large, it is not necessary to separate the training set. The value of s determines how many previous instances is going to be stored to predict the upcoming instances. A larger s requires more storage. So, for data streams with a reoccurring concept problem, we suggest a larger s. In addition, a larger s means we need a larger storage, so $s \times w$ is also limited by the device. If w is large, it may not be able to choose a large s anymore. In addition, c determines how many predictors are ensembled. Therefore, for data streams that are difficult to predict, we suggest a larger c. The k for kNN search can also be determined by discretization controlling method [16]

In general, SEGA is robust on these two parameters on most tested data streams according to the results shown in 4 and 5. For the data streams of Sensor3, Sensor8, Sensor20 and Sensor46, SEGA obtain better results with smaller w, which denotes that drift occurs at a relatively high frequency in these four data streams.

The computation complexity of SEGA is determined by the size of the segments (w) and the number of segments in each training set (s). The complexity of computing SSD is $O(s \times w^2)$ for every w instances. In each learning process, the complexity is $O((s-1) \times w)$ when computing the drift gradient of δ_Q and is less than $O((s-1)^2 w^2)$ when computing the drift gradient of δ_P. Given the size of training data as N, in each learning process, the computation complexity is upper bounded by $O(N^2)$ if SEGA updates δ_P and the complexity is upper bounded by $O(N)$ if the updates of δ_P are skipped, which is competitive SAM_kNN($O(NL_{max}^2 log\frac{L_{max}}{L_{min}})$) and faster than FUZZ-CARE($\sum_{k=2}^{K} O(NI + 2k)$).

The run time of SEGA implemented in Python is listed in Table XI and XII. In addition, we also provide the run time of other baseline methods implemented in Java. These results are just for a reference rather than for comparison, as most baseline method's Java code applies parallel. The execution time of SEGA could be shortened if parallel computing is involved to shorten the run time of for-loop. Meanwhile, although SEGA uses kNN search the neighbors for updating drift-gradient, and a kNN predictor for classification tasks, we do these two procedures separately, which has repeated computation. However, this makes SEGA more flexible when choosing predictor and computing the distance matrix.

V. Conclusion and Future Studies

Concept drift is one of the most challenging problems for learning data streams, where the data distribution changes over time. In this paper, the concept drift definition is revised by adding a constraint, which can distinctly distinguish drift from outliers.

To solve the concept drift problem, this paper proposed an online adaptation method, called SEGA, to predict labels for data streams of both regression and classification tasks. Instead of using the whole training data to retrain predictors, which is common in most recent research on concept drift adaptation, SEGA train and retrain predictors on selected segments of the training data. In SEGA, we proposed a sequentially updated statistic, drift-gradient, to select the optimal segments when every new instance arrives. In this way, SEGA can overcome the delay of the informed drift adaptation method, as well as the instability of the blind drift adaptation method. Our SEGA method is validated by experiments on 30 data including synthetic or real-world data streams of classification or regression tasks. The consistent performance shows that SEGA is able to solve various types of drift under different situations.

SEGA implements adaptation based on the value of drift-gradient. Compared to the exiting statistics in this field, drift-gradient responds to drift faster. Here, drift-gradient denotes the value of the increase of distributional discrepancy from $t-1$ to t, rather than the value of distributional discrepancy at t. An intuitive example of their difference is that we refer the concept as a force, the *distributional discrepancy* as velocity, and drift-gradient as acceleration. If the force changes (drift occurs), it is reflected immediately in acceleration, while velocity just starts to change.

In SEGA, the segments are simply determined by the time windows of instances. However, this segment does not comply with real-world data streams. Therefore, if we can segment the training set more precisely, it will help further improve the performance of SEGA. How to choose the best ensemble predictor, the size and the number of segments in the training set for different data streams is not discussed in this paper. They can be selected and optimized by introducing other techniques, such as cross validation on the original training set, or adaptive adaptation updating these parameters. To overcome the above mentioned drawbacks in current version of SEGA, our future work will add adaptive settings in SEGA to make it more efficient. Meanwhile, we will try to apply SEGA in a multiple streams case. When multiple streams have inner correlations, the drift-gradient needs to include those correlations. Therefore, the version of drift-gradient and SEGA for multiple streams is also promising as future research.

Acknowledgments

The work presented in this paper was supported by the Australian Research Council (ARC) under the Discovery Project DP190101733.

References

[1] M. Sayed-Mouchaweh and E. Lughofer, *Learning in non-stationary environments: methods and applications*. Springer, New York, NY, 2012.

[2] M. Jaworski, P. Duda, and L. Rutkowski, "New splitting criteria for decision trees in stationary data streams," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2516–2529, 2018.

[3] G. S. Gurjar and S. Chhabria, "A review on concept evolution technique on data stream," in *2015 International Conference on Pervasive Computing (ICPC)*. Pune, India, Jan. 8-10: IEEE, 2015, pp. 1–3.

[4] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 4802–4821, 2018.

[5] J. C. Schlimmer and R. H. Granger, "Beyond incremental processing: Tracking concept drift." in *the 5th AAAI Conference on Artificial Intelligence*, Philadelphia Pennsylvania, USA, Aug. 11-15, 1986, pp. 502–507.

[6] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving systems*, vol. 9, no. 1, pp. 1–23, 2018.

[7] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[8] N. Lu, J. Lu, G. Zhang, and R. L. De Mantaras, "A concept drift-tolerant case-base editing technique," *Artificial Intelligence*, vol. 230, pp. 108–133, 2016.

[9] G. Boracchi, D. Carrera, C. Cervellera, and D. Maccio, "Quanttree: Histograms for change detection in multivariate data streams," in *the 35th International Conference on Machine Learning*, Stockholm, Sweden, Jul.10-15, 2018, pp. 638–647.

[10] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, 2018, doi:10.1109/TKDE.2018.2876857.

[11] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1179–1199, 2018.

[12] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer, "Concept drift detection through resampling," in *the 31st International Conference on Machine Learning, Beijing, China, Jun. 21-26*, 2014, pp. 1009–1017.

[13] B. S. Parker and L. Khan, "Detecting and tracking concept class drift and emergence in non-stationary fast data streams," in *29th AAAI Conference on Artificial Intelligence*, Austin Texas, USA, Jan 25-30, 2015, pp. 2908–2913.

[14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys*, vol. 46, no. 4, p. 44, 2014.

[15] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *IEEE 32nd International Conference on Data Engineering*. Helsinki, Finland, May. 16-20: IEEE, 2016, pp. 481–492.

[16] A. Liu, J. Lu, F. Liu, and G. Zhang, "Accumulating regional density dissimilarity for concept drift detection in data streams," *Pattern Recognition*, vol. 76, pp. 256–272, 2018.

[17] S. Yu, X. Wang, and J. C. Principe, "Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels," *arXiv preprint arXiv:1806.10131*, 2018.

[18] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[19] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2014.

[20] A. Bifet, W. Fan, C. He, Q. Jianfeng, Z. Jianfeng, and G. Holmes, "Extremely fast decision tree mining for evolving data streams," in *the 23rd SIGKDD Conference on Knowledge Discovery and Data Mining*, Halifax, NS, Canada, Aug. 1317., 2017, pp. 1733–1742.

[21] D. M. dos Reis, P. Flach, S. Matwin, and G. Batista, "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California, USA, Aug. 13-17: ACM, 2016, pp. 1545–1554.

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

[23] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE transactions on knowledge and data engineering*, vol. 24, no. 4, pp. 619–633, 2012.

[24] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Computing Surveys*, vol. 50, no. 2, p. 23, 2017.

[25] M. Pratama, J. Lu, E. Lughofer, G. Zhang, and M. J. Er, "An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 5, pp. 1175–1192, 2017.

[26] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2014.

[27] Y. Lu, Y.-m. Cheung, and Y. Y. Tang, "Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift." in *the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, Aug.19-25, 2017, pp. 2393–2399.

[28] Y. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept drift adaptation by exploiting historical knowledge," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 4822–4832, 2018.

[29] A. Haque, L. Khan, and M. Baron, "SAND: Semi-supervised adaptive novel class detection and classification over data stream," in *the 30th AAAI Conference on Artificial Intelligence*, Phoenix Arizona, USA, Feb. 12-17, 2016, pp. 1652–1658.

[30] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "A new method for data stream mining based on the misclassification error," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 5, pp. 1048–1059, 2015.

[31] J. Shao, Z. Ahmadi, and S. Kramer, "Prototype-based learning on concept-drifting data streams," in *the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, USA, Aug.24-27: ACM, 2014, pp. 412–421.

[32] C. Alippi, G. Boracchi, and M. Roveri, "Hierarchical change-detection tests," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 2, pp. 246–258, 2017.

[33] L. Bu, C. Alippi, and D. Zhao, "A pdf-free change detection test based on density difference estimation," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 2, pp. 324–334, 2018.

[34] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, "Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances," *Information Sciences*, vol. 355, pp. 127–151, 2016.

[35] A. Shaker and E. Lughofer, "Self-adaptive and local strategies for a smooth treatment of drifts in data streams," *Evolving Systems*, vol. 5, no. 4, pp. 239–257, 2014.

[36] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, Aug. 19-25, 2017, pp. 2280–2286.

[37] A. Liu, "Concept drift datasets," https://github.com/Anjin-Liu/ConceptDriftDatasets/tree/master/Synthetic, 2019.

[38] Y. Song, "FUZZ-CARE," https://github.com/songyiliao/FUZZ-CARE, 2019.

[39] K. Yeon, M. S. Song, Y. Kim, H. Choi, and C. Park, "Model Averaging via Penalized Regression for Tracking Concept Drift," *Journal of Computational and Graphical Statistics*, vol. 19, no. 2, pp. 457–473, 2010.

[40] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[41] E. Ikonomovska, J. Gama, B. Ženko, and S. Džeroski, "Speeding-up hoeffding-based regression trees with options," in *the 28th International Conference on Machine Learning*. Bellevue Washington, USA, Jun. 28- Jul. 2: Citeseer, 2011, pp. 537–544.

[42] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data mining and knowledge discovery*, vol. 23, no. 1, pp. 128–168, 2011.

[43] J. Duarte, J. Gama, and A. Bifet, "Adaptive model rules from high-speed data streams," *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 3, p. 30, 2016.

[44] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank, "Fast perceptron decision tree learning from evolving data streams," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Hyderabad, India, Jun. 21-24, 2010, pp. 299–310.

[45] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

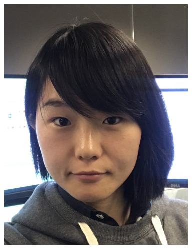
[46] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Joint European conference on machine learning and knowledge discovery in databases*. Barcelona, Spain, Sep. 19-23: Springer, 2010, pp. 135–150.

[47] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift," in *IEEE 16th International Conference on Data Mining (ICDM)*. Barcelona, Spain, Dec 12-15: IEEE, 2016, pp. 291–300.

[48] A. Shaker and E. Hüllermeier, "Iblstreams: a system for instance-based classification and regression on data streams," *Evolving Systems*, vol. 3, no. 4, pp. 235–249, 2012.

[49] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[50] T. Pohlert, "The pairwise multiple comparison of mean ranks package (PMCMR)," *R package*, pp. 2004–2006, 2014.



Yiliao Song (S'17) received a M.S. in probability and statistics in mathematics from the School of Mathematics and Statistics, Lanzhou University, China, in 2015. She is working toward a Ph.D. with the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. Her research interests include regression, prediction, concept drift and data stream mining. She has published 17 papers related to concept drift, and data stream prediction.

Jie Lu (F'18) is a Distinguished Professor and the Director of Australian Artificial Intelligence Institute (AAII) at the University of Technology Sydney, Australia. She is also an IFSA Fellow and Australian Laureate Fellow. She received a PhD degree from Curtin University, Australia, in 2000. Her main research expertise is in transfer learning, concept drift, decision support systems and recommender systems. She has been awarded 10 Australian Research Council (ARC) discovery grants and led 15 industry projects. She has published over 450 papers in IEEE transactions and other journals and conferences, supervised 40 PhD students to completion. She serves as Editor-In-Chief for Knowledge-Based Systems (Elsevier) and Editor-In-Chief for International Journal on Computational Intelligence Systems (Atlantis). She has delivered 27 keynote speeches at IEEE and other international conferences and chaired 15 international conferences. She has received the UTS Medal for Research and Teaching Integration (2010), the UTS Medal for research excellence (2019), the IEEE Transactions on Fuzzy Systems Outstanding Paper Award (2019), the Computer Journal Wilkes Award (2018), and the Australian Most Innovative Engineer Award (2019).

Anjin Liu (M'17) is a Postdoctoral Research Associate in the A/DRsch Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. He received the BIT degree (Honour) at the University of Sydney in 2012. His research interests include concept drift detection, adaptive data stream learning, multi-stream learning, machine learning and big data analytics.

Haiyan (Helen) Lu (SM'15) received the B.Eng. and M.Eng. degrees in electrical engineering from the Harbin SEGAtitute of Technology, Harbin, China, in 1985 and 1988, respectively, and the Ph.D. degree in engineering from the University of Technology Sydney, Ultimo, NSW, Australia, in 2002.

She is currently an associate professor with the School of Computer Science, University of Technology Sydney, Australia. She is a core member of the Decision Systems & e-Service Intelligence Lab in the Centre for Artificial Intelligence. She has published three book chapters, 53 refereed international journal articles and 80 refereed conference papers. Her research interests include computational intelligence with focus on evolutionary optimization algorithms, time series forecasting, ontology, recommendation techniques and their applications in business and engineering, especially in smart cities. She is a senior member of IEEE.

Guangquan Zhang is an Australian Research Council (ARC) QEII Fellow, Associate Professor and the Director of the Decision Systems and e-Service Intelligent (DeSI) Research Laboratory at the Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. He received his Ph.D in applied mathematics from Curtin University, Australia, in 2001. From 1993 to 1997, he was a full Professor in the Department of Mathematics, Hebei University, China. His main research interests lie in the area of fuzzy multi-objective, bilevel, and group decision making, fuzzy measure, and machine learning. He has published six authored monographs, five edited research books, and over 450 papers including 240 refereed journal articles. Dr. Zhang has won nine ARC Discovery Project grants and many other research grants, supervised 30 PhD students to completion. He has served as a Guest Editor for five special issues of IEEE Transactions and other international journals.

Appendix A
Derivation step from Symmetric Degree (SD) to Segmented Symmetric Degree(SSD)

According to (5), we have

$$\bar{d}_{P,Q}(k) = \frac{1}{N_P}\sum_{u\in P_1}\left(\frac{N_{u,P}(k)}{N_P} - \frac{N_{u,Q}(k)}{N_Q}\right) + \frac{1}{N_P}\sum_{u\in P_2}\left(\frac{N_{u,P}(k)}{N_P} - \frac{N_{u,Q}(k)}{N_Q}\right) + \frac{1}{N_Q}\sum_{v\in Q}\left(\frac{N_{v,Q}(k)}{N_Q} - \frac{N_{v,P}(k)}{N_P}\right). \tag{20}$$

As $N_{u,P}(k) + N_{u,Q}(k) = k$, the first two terms in $\bar{d}_{P,Q}(k)$ can be written by

$$\frac{1}{N_P}\sum_{u\in P_1}\left(\frac{N_{u,P}(k)}{N_P} - \frac{N_{u,Q}(k)}{N_Q}\right) = \frac{1}{N_P}\sum_{u\in P_1}\left(\frac{k - N_{u,Q}(k)}{N_P} - \frac{N_{u,Q}(k)}{N_Q}\right), \tag{21}$$

$$\frac{1}{N_P}\sum_{u\in P_2}\left(\frac{N_{u,P}(k)}{N_P} - \frac{N_{u,Q}(k)}{N_Q}\right) = \frac{1}{N_P}\sum_{u\in P_2}\left(\frac{k - N_{u,Q}(k)}{N_P} - \frac{N_{u,Q}(k)}{N_Q}\right). \tag{22}$$

The third term in (20) is

$$\frac{1}{N_Q}\sum_{v\in Q}\left(\frac{N_{v,Q}(k)}{N_Q} - \frac{N_{v,P}(k)}{N_P}\right) = 2\times\frac{1}{2N_Q}\sum_{v\in Q}\frac{N_{v,Q}(k)}{N_Q} - \frac{1}{N_Q}\sum_{v\in Q}\frac{N_{v,P}(k)}{N_P}. \tag{23}$$

The term $N_{v,P}(k)$ represents the number of v's k nearest neighbors that are from P. As $P = P_1 \cup P_2$ and $P_1 \cap P_2 = \emptyset$, $N_{v,P}(k) = N_{v,P_1}(k) + N_{v,P_2}(k)$, we have

$$\frac{1}{N_Q}\sum_{v\in Q}\left(\frac{N_{v,Q}(k)}{N_Q} - \frac{N_{v,P}(k)}{N_P}\right) = \frac{1}{2N_Q}\sum_{v\in Q}\frac{N_{v,Q}(k)}{N_Q} + \frac{1}{2N_Q}\sum_{v\in Q}\frac{N_{v,Q}(k)}{N_Q} - \frac{1}{N_Q}\sum_{v\in Q}\frac{N_{v,P_1}(k)}{N_P} - \frac{1}{N_Q}\sum_{v\in Q}\frac{N_{v,P_2}(k)}{N_P}. \tag{24}$$

Substituting (21), (22) and (24) into (20), we have $\bar{d}_{P,Q}(k) = ssd_{P_1,Q}(k) + ssd_{P_2,Q}(k)$.

APPENDIX B
SUPPLEMENTARY EXPERIMENTAL RESULTS

TABLE IX
RESULTS OF SYNTHETIC DATA OVER BASELINE METHODS(REGRESSION)

	ORTO	FIMTDD	metaAMR	AMR	Per	FUZZ-CARE_Linear	SEGA_linear
Non-Drift	1.24	2.09	0.82	0.82	3.75	1.18	0.81
Virt-Drift	1.31	6.19	0.80	0.80	7.24	1.30	0.79
Sudd-Drift	3.65	4.69	8.44	8.44	4.86	3.34	2.72
Incre-Drift	2.84	0.82	8.46	8.46	0.81	4.03	2.31
Rec-Drift-Grad	8.24	0.82	8.37	8.37	0.80	2.34	1.27
Rec-Drift-Mix	6.12	2.18	5.91	5.91	3.38	2.13	1.68

TABLE X
RESULTS OF SYNTHETIC DATA OVER BASELINE METHODS(CLASSIFICATION)

	kNN	LevBag_KNN	IBLStream_KNN	SAM_kNN	Learn++NSE_KNN	SlidWin_kNN	**SEGA**_kNN
SEAa	80.95	78.55	79.42	85.39	82.79	81.85	84.41
SEAg	80.95	78.03	79.37	85.04	82.64	81.79	84.60
HYPER	77.38	68.11	77.94	79.89	72.68	78.34	78.67
AGRs	50.51	62.77	64.00	49.12	52.85	61.66	61.99
AGRg	50.51	73.24	69.00	49.15	52.98	61.59	62.49
RTG	73.91	58.65	65.79	59.59	64.10	73.85	65.69

TABLE XI
RUN-TIME ON REGRESSION REAL-WORLD DATA STREAMS (S CPU-TIME)

Data Streams	ORTO	FIMTDD	metaAMR	AMR	Per	SEGA
CCPP	2.11	1.02	29.31	3.03	1.00	58.71
house	1.02	1.02	12.09	1.01	1.02	158.98
sensor3	1.01	1.02	28.30	2.03	1.02	170.47
sensor8	1.02	1.01	8.06	1.01	1.02	50.50
sensor20	1.01	1.02	16.14	2.02	1.02	102.61
sensor46	1.02	1.01	36.31	2.05	1.02	193.75
SMEAR	1.02	1.01	77.68	8.06	2.02	540.83
Solar	1.09	1.00	18.23	2.03	1.02	99.57

SEGA is implemented in Python, while other methods are implemented in Java by MOA

TABLE XII
RUN-TIME ON CLASSIFICATION REAL-WORLD DATA STREAMS (S CPU-TIME)

CPU(s)	ADWIN-ARF	NN-DVI$_{kNN}$	LevBag$_{kNN}$	OnlineAUE	IBLStream	Learn++NSE	SlidWin$_{kNN}$	**SEGA**$_{kNN}$
Elec	11.20	374.13	598.71	4.08	8.98	6.20	2.97	173.50
Weather	4.01	183.64	329.23	1.00	37.09	2.02	6.39	82.58
Spam	6.01	4899.40	11331.10	7.56	1131.02	5.02	33.09	343.20
Airline	355.19	5754.27	7336.86	196.15	1314.86	852.05	213.51	2295.45
Covertype	123.04	86436.71	48499.82	112.04	2344.83	2357.61	333.53	3678.54
Usenet1	1.00	132.66	377.24	1.00	4.59	1.01	0.19	1.23
Usenet2	1.01	124.08	382.16	1.00	5.11	1.01	0.18	1.24

SlidWin, SEGA$_kNN$ are implemented in Python, while other methods are implemented in Java by MOA