A Drift Region-Based Data Sample Filtering Method

Fan Dong, Jie Lu, *Fellow, IEEE*, Yiliao Song, *Member, IEEE*, Feng Liu, *Member, IEEE*, Guangquan Zhang

***Abstract*—Concept drift refers to changes in the underlying data distribution of data streams over time. A well-trained model will be outdated if concept drift occurs. Once concept drift is detected, it is necessary to understand where drift occurs to support the drift adaptation strategy and effectively update outdated models. This process, called drift understanding, has rarely been studied in this area. To fill this gap, this paper develops a drift region-based data sample filtering method to update the obsolete model and track the new data pattern accurately. The proposed method can effectively identify the drift region and utilize information on the drift region to filter the data sample for training models. Theoretical proof guarantees the identified drift region converges uniformly to the real drift region as the sample size increases. Experimental evaluations based on four synthetic datasets and two real-world datasets demonstrate our method improves the learning accuracy when dealing with data streams involving concept drift.**

***Index Terms*—concept drift, data stream, nonstationary environment, machine learning**

I. Introduction

Real-world applications generate a tremendous amount of streaming data, and learning from data streams is among the most vital contemporary fields in machine learning and data mining [1], [2]. In streaming data, the data distribution may change over time as new data arrive [3], [4]. This problem is termed as *concept drift*, such as the recognition pattern shifts when new images are added to the database [5], [6], changes in user interest in shopping behaviors, and the emergence of new types of spam in email filtering systems [7]. Once concept drift occurs, the patterns induced from past data may not be relevant to the latest data [8]. Learning under concept drift becomes a big challenge when applying machine learning techniques into data stream mining applications.

A typical framework of learning under concept drift (Fig. 1) consists of three main modules: 1) *concept drift detection* detects when drift occurs in the data stream; 2) *concept drift understanding* retrieves useful information where concept drift has occurred, such as identifying the drift region; and 3) *concept drift adaptation* designs a strategy to update the learning model by incorporating the outputs of concept drift detection and concept drift understanding to track the newly arrived concepts promptly. A clear taxonomy of concept drift adaptation was given by [9].

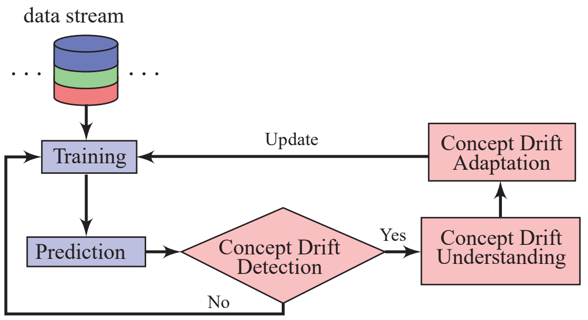


Fig. 1: A framework of learning under concept drift.

So far, limited research focuses on the concept drift understanding module. Most existing concept drift techniques update the learning model when drift is detected [10]. Once a concept drift is detected, it is necessary to understand the drift region where drift has occurred; a process known as drift understanding. Concept drift understanding will directly support the drift adaptation strategy to update the outdated models to fit newly arrived data.

To fill this research gap, we propose a drift region-based data sample filtering method to deal with the concept drift problem by understanding where the drift occurred. The proposed method uses information in the drift region to filter the data sample. This filtered data sample represents the latest pattern and therefore, will help to update the learning model accurately.

Specifically, the proposed drift region-based data sample filtering method consists of three modules: 1) the drift detection module is realized by a competence model-based drift detection (CM-DD) method [11] that monitors the data distribution changes of the data stream; 2) the drift understanding module utilizes the cumulative distribution functions and the degree of drift to identify the drift region; 3) the drift adaptation module updates the data sample according to the output of drift understanding module and trains models on the updated data sample.

The main contributions in this paper are listed as follows.

1) We extend the theoretical foundation of the competence model-based drift detection (CM-DD) method [11] so that it can combine with the drift understanding module, in principal.

2) A competence model-based drift region identification (CM-DRDI) method is proposed to understand drift which can effectively identify the drift region.

3) In CM-DRDI, we prove the upper and lower bounds of competence-based discrepancy density estimation (CDDE)

F. Dong, J. Lu, Y. Song, F. Liu and G. Zhang are with the Decision Systems and e-Service Intelligence Laboratory, The Australian Artificial Intelligence Institute, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: Fan.Dong@uts.edu.au; Jie.Lu@uts.edu.au; Yiliao.Song@uts.edu.au; Feng.Liu@uts.edu.au; Guangquan.Zhang@uts.edu.au.).

The code and relevant data for this paper can be found in https://code.research.uts.edu.au/120074/cm-drift.

by ensuring the output of concept drift understanding is reliable. The theoretical proof guarantees that as the sample size increases, the identified drift region converges with the real concept drift region uniformly.

4) Based on CM-DRDI, We develop a competence model-based drift region filtering (CM-DRIFT) method for drift adaptation. CM-DRIFT updates the data sample by removing obsolete data in the drift region from the data sample and adding new valuable data into the data sample. Therefore, CM-DRIFT guarantees that the model trained with the updated data sample can always represent the new concept.

Comprehensive comparison experiments are conducted on on four synthetic datasets and two real-world datasets to evaluate the proposed modules. The rest of this paper is organized as follows. Section II discusses the related work. Section III presents our proposed drift region-based data sample filtering method. Section IV describes and discusses the results of the experimental evaluation and Section V concludes this study with suggestions for future work.

II. Related work

Incremental algorithms partially update the existing learning model rather than retraining an entire learner when concept drift has occurred [12]. These approaches are arguably more efficient when drift only occurs in local regions. Many models in this category are based on a decision tree algorithm because decision trees have the ability to examine and adapt to each sub-region separately. CVFDT [13] is an online decision tree algorithm that can handle concept drift. In CVFDT, a sliding window is maintained to hold the latest data. An alternative sub-tree is trained based on the window, and its performance is monitored. If the alternative sub-tree outperforms its original counterpart, it is used for future prediction and the original obsolete sub-tree is pruned. VFDTc [14] is another attempt to improve VFDT with several enhancements: the ability to handle numerical attributes; the application of naive Bayes classifiers in tree leaves; and the ability to detect and adapt to concept drift. The Hoeffding tree [15] stores old information in root nodes and new information in the leaves. By comparing the distribution of the errors in the leaf nodes with the upper nodes, Hoeffding trees can reveal whether a drift has occurred. Similar implementations have been adopted in HAT [16] and FIMT-DD [17].

KNN-based algorithms have emerged in recent years [18]. KNN-PAW [19] integrates probabilistic adaptive windowing in its model and eliminates instances in the kNN model so as to give new instances greater weight while de-emphasising older ones. SAM-kNN [20] maintains two kNN models and separates the instances into the two models – one for the old concept, one for the new one. Only one model is used to make predictions and which one depends on the given situation NEFCS [21] is another adaptation method based on kNN.

Informed adaptation strategies only conduct adaptation when drift is detected; hence, the informed part of the nomenclature. As such, a concept drift detection method is an integral part of any informed drift adaptation approach. One of the most commonly used concept drift detection algorithms is the "drift detection method" [10]. It monitors the online error-rate of the base classifier to determine whether there are changes in the new incoming data. DDM can work independently of the base classifier because it only needs information on whether the base classifier has classified the data instance correctly. Lu et al. [11] proposed a competence model-based drift detection that uses competence-based empirical distance to show the difference between two data windows. Their method confirms that a drift has occurred by verifying whether the empirical competence-based distance is sufficiently large through a permutation test. Liu et al. [22] proposed a regional density estimation-based drift detection method NN-DVI, which is sensitive to small regional drift. Similar distribution-based drift detection methods are: fuzzy competence model drift detection [23], equal density estimation [24], and local drift degree-based density synchronized drift adaptation [25].

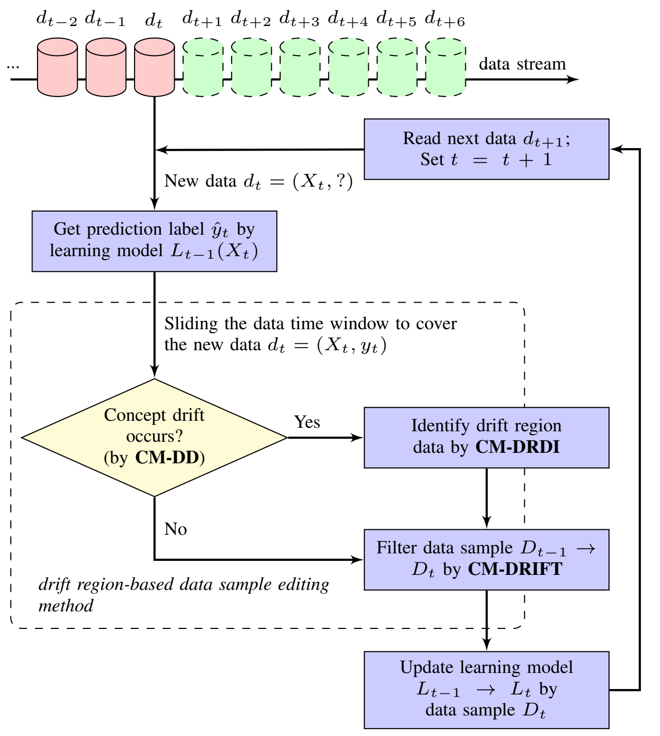
In spite of their high computation costs, adaptive ensembles are still drawing the interests of the research community. Adaptive ensembles handle concept drift by extending classical ensemble methods or by following specific adaptive voting rules. Dynamic Weighted Majority (DWM) [26] is an ensemble method that is able to adapt to drifts with a simple set of weighted voting rules. It manages base classifiers according to the performance of both the individual classifiers and the global ensemble. If the ensemble incorrectly predicts an instance, DWM will train a new base classifier and add it to the ensemble. If a base classifier incorrectly predicts an instance, DWM reduces its weight by a factor. When the weight of a base classifier drops below a user-defined threshold, DWM removes it from the ensemble. A similar technique is SEA [27]. SEA maintains a fixed-size ensemble of classifiers, and each classifier is trained from a batch of a data stream. When a new batch of data becomes available, a new classifier is added into the ensemble and the worst classifier is eliminated. Learn++.NSE [28] has the advantage of being able to handle any type of concept drift. Learn++.NSE does not store historical data, only the latest batch of data and the base classifiers trained by each batch of data. Underperforming classifiers can be reactivated or deactivated as needed by adjusting their weights. Other adaptive ensemble strategies have been applied to handle concept drift, such as hierarchical ensemble structure [29], AUE2 [30], and DTEL [31].

III. Methodology description

This method is inspired by the idea that: when there is no concept drift, current data instances in the data sample can help to identify noise and improve classification accuracy. On the contrary, when concept drift occurs, new data instances are more suitable to represent the new concept, and the obsolete data instances should be removed from the data sample.

Drift region data refers to the obsolete instances, and the core technique of our method is to accurately identify the drift region data.

An overview of our method is shown in Fig. 2. For each new available data d_t in the data stream, the latest learning model L_{t-1} classifies the new data and obtains the prediction. After that, we assume the new data are labeled and ready for



d_{t-2} d_{t-1} d_t d_{t+1} d_{t+2} d_{t+3} d_{t+4} d_{t+5} d_{t+6}
...
data stream
Read next data d_{t+1}; Set $t = t+1$
New data $d_t = (X_t, ?)$
Get prediction label $\hat{y}_t$ by learning model $L_{t-1}(X_t)$
Sliding the data time window to cover the new data $d_t = (X_t, y_t)$
Concept drift occurs? (by CM-DD)
Yes
Identify drift region data by CM-DRDI
No
Filter data sample $D_{t-1} \to D_t$ by CM-DRIFT
drift region-based data sample editing method
Update learning model $L_{t-1} \to L_t$ by data sample D_t

Fig. 2: Overview of data stream classification using drift region-based data sample editing method.

further processing. The data time window slides and covers the new data, and then CM-DD is performed to detect if concept drift has occurred. If concept drift has occurred, CM-DRDI uses the outputs of CM-DD to identify drift region data, and then CM-DRIFT utilizes the outputs of CM-DRDI to filter the existing data sample. If no concept drift has occurred, CM-DRIFT adjusts the data sample and enhances the ability of the data sample to represent the new concept. After data sample is filtered, the learning model updates itself with the latest data sample D_t. The last step is to read the next available data instance.

In the next subsections, the three modules in Fig. 2 will be explained. Before that, some of the notations for data are listed below:

- $\mathcal{X} = \{X_1, \dots, X_n\}$ where X_i is a d-dimensional data instance
- $\mathcal{X}_0 = \{X_1, \dots, X_6\}$ is an example of $\mathcal{X}$
- w_p denotes the previous time window of data
- w_c denotes the current time window of data

A. Competence model-based drift detection—CM-DD

The essence of CM-DD is to monitor if the data distributions between the previous time window of data w_p and the current time window of data w_c have significant difference. Since the true distributions of w_p and w_c are unknown, CM-DD estimates their empirical data distributions $\hat{F}_t$ and $\hat{F}_{t'}$ instead. To obtain the empirical data distribution, CM-DD partitions the union set $w_p \cup w_c$ into a group of overlapped $\mathrm{RelatedSet}$ of competence models, and then estimates their empirical data distributions $\hat{F}_t$ and $\hat{F}_{t'}$ using the *competence-based density vector* and *competence-based empirical vector*, respectively. The difference between $\hat{F}_t$ and $\hat{F}_{t'}$ is measured

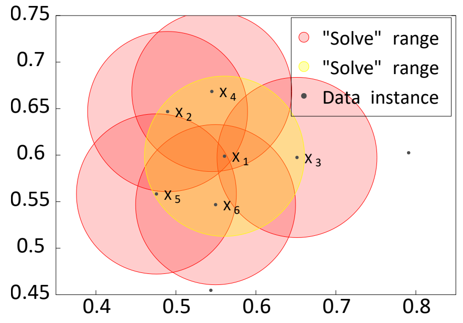


Fig. 3: An example of competence model.

by the *competence-based empirical distance* $\mathrm{d}_{\mathcal{E}}$. The relevant definitions of CM-DD are given as follows.

Definition 1. Coverage Set. For any $X_i \in \mathcal{X}$, the coverage set [32] of X_i is defined as:

$$\mathcal{C}(X_i) = \{X_j \in \mathcal{X} : \mathrm{Solve}(X_i, X_j)\}, \tag{1}$$

where $\mathrm{Solve}(X_i, X_j)$ indicates a Solve rule that X_i can be retrieved and adapted to solve X_j. In a classification task, the Solve rule is a set to meet following conditions: 1) X_i and X_j have the same class label; 2) X_k is the closest data of X_i which has a different class label to X_i; 3) the distance between X_i and X_j is less than the distance between X_i and X_k.

Example 1. Given $\mathcal{X}_0 = \{X_1, \dots, X_6\}$ a set of data instances, the Solve rule is defined as two data that mutually solve each other if the Euclidean distance between the two data is less than 0.1, as shown in Fig. 3. The color-filled circles highlight the Solve range of the data. $X_1, X_2, X_3, X_4, X_5, X_6$ are located within the Solve range of X_1, which is highlighted by the yellow circle. According to Definition 1, $\mathcal{C}(X_1) = \mathcal{X}_0$.

Definition 2. Reachability Set. For any $X_i \in \mathcal{X}$, the Reachability Set [32] of X_i is defined as:

$$\mathcal{R}(X_i) = \{X_j \in \mathcal{X} : \mathrm{Solve}(X_j, X_i)\} \tag{2}$$

Example 2. In Fig. 3, X_1 is located within the Solve range of $X_1, X_2, X_3, X_4, X_5, X_6$, Therefore, $\mathcal{R}(X_1) = \mathcal{X}_0$.

Definition 3. Related Set. For any $X_i \in \mathcal{X}$, the related Set [33] of X_i is defined as:

$$\mathcal{R}_s(X_i) = \mathcal{C}(X_i) \cup \mathcal{R}(X_i) \tag{3}$$

Example 3. In Fig. 3, $\mathcal{R}_s(X_1) = \mathcal{C}(X_1) \cup \mathcal{R}(X_1) = \mathcal{X}_0$.

Definition 4. Related Closure. For any $X_i \in \mathcal{X}$, the related closure [11] of X_i is defined as:

$$\begin{aligned}\Re(X_i) &= \{\mathcal{R}_s(X_j) : X_i \in \mathcal{R}_s(X_j), j = 1, \dots n\} \\ &= \{r_1, r_2, \dots, r_m, m \leq n\}\end{aligned} \tag{4}$$

Moreover, for a subset $S \subseteq \mathcal{X}$, the related closure of S is defined as:

$$\Re(S) = \cup_{X_i \in S} \Re(X_i) \tag{5}$$

Example 4. In Fig. 3, $\mathcal{R}_s(X_1)$ is given in example 3, $\mathcal{R}_s(X_2) = \{X_1, X_2, X_4\}$, $\mathcal{R}_s(X_3) = \{X_1, X_3\}$, and $\mathcal{R}_s(X_4) = \{X_1, X_2, X_4\}$. Given $S = \{X_2, X_3\}$, the related closure of S is computed as $\Re(S) = \Re(X_2) \cup \Re(X_3)$. $\Re(X_2) = \{\mathcal{R}_s(X_1), \mathcal{R}_s(X_2), \mathcal{R}_s(X_4)\} = \{\mathcal{X}_0, \{X_1, X_2, X_4\}\}$, and $\Re(X_3) = \{\mathcal{R}_s(X_1), \mathcal{R}_s(X_3)\} = \{\mathcal{X}_0, \{X_1, X_3\}\}$. Therefore, $\Re(S) = \{\mathcal{X}_0, \{X_1, X_2, X_4\}\}, \{X_1, X_3\}$.

Definition 5. Competence-based Density Vector. For any $X_i \in \mathcal{X}$, the competence-based density vector of X_i is defined as:

$$\nu(X_i) = \left(\frac{|\{X_i\} \cap r_1|}{|\Re(X_i)|}, \frac{|\{X_i\} \cap r_2|}{|\Re(X_i)|}, \dots, \frac{|\{X_i\} \cap r_m|}{|\Re(X_i)|}\right) \tag{6}$$

$\nu(X_i)$ is a $1 - by - m$ vector. It represents the density probability of X_i in the competence model-based measurable space. In addition, for $\nu(X_i) = (w_1, w_2, \dots, w_m)$, $\sum_{i=1}^m w_i = 1$.

Example 5. In Fig. 3, $\mathcal{R}_s(X_1) = \mathcal{X}_0$; $\mathcal{R}_s(X_2) = \mathcal{R}_s(X_4) = \{X_1, X_2, X_4\}$; $\mathcal{R}_s(X_3) = \mathcal{R}_s(X_6) = \{X_1, X_3\}$. The related closure $\Re(\mathcal{X}) = \{\mathcal{R}_s(X_1), \mathcal{R}_s(X_2), \mathcal{R}_s(X_3), \mathcal{R}_s(X_5)\}$, and the $\nu(X_2)$ is calculated by:

$$\nu(X_2) = (\frac{|\{X_2\} \cap \mathcal{R}_s(X_1)|}{|\Re(X_2)|}, \frac{|\{X_2\} \cap \mathcal{R}_s(X_2)|}{|\Re(X_2)|}, \frac{|\{X_2\} \cap \mathcal{R}_s(X_3)|}{|\Re(X_2)|}, \frac{|\{X_2\} \cap \mathcal{R}_s(X_5)|}{|\Re(X_2)|}) = (\frac{1}{2}, \frac{1}{2}, \frac{0}{2}, \frac{0}{2}) = (\frac{1}{2}, \frac{1}{2}, 0, 0)$$

Definition 6. Competence-based Empirical Vector. Given $S \subseteq \mathcal{X}$, the competence-based empirical vector of S is defined as:

$$\mathcal{E}(S) = \frac{1}{|S|} \sum_{X_i \in S} \nu(X_i) \tag{7}$$

$\mathcal{E}(S)$ is essentially an average of $\nu(X_i)$. The sum of each elements in $\mathcal{E}(S)$ equals 1. This empirical vector $\mathcal{E}(S)$ can be treated as the discrete probability distribution of the data instances over competence model-based measurement space.

Example 6. In Fig. 3, given $S = \{X_1, X_2, X_3\}$, $\nu(X_1)$ is calculated by:

$$\nu(X_1) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}), \nu(X_2) = (\frac{1}{2}, \frac{1}{2}, 0, 0), \nu(X_3) = (\frac{1}{2}, 0, \frac{1}{2}, 0)$$
$$\mathcal{E}(S) = \frac{1}{3}(\nu(X_1) + \nu(X_2) + \nu(X_3)) = (\frac{5}{12}, \frac{1}{4}, \frac{1}{4}, \frac{1}{12})$$

Definition 7. Competence-based Empirical Distance. Given two subsets $S_1, S_2 \subset X$, $\mathcal{E}(S_1) = (p_1, \dots, p_m)$ and $\mathcal{E}(S_2) = (q_1, \dots, q_m)$ are their respective competence-based empirical vectors, the competence-based empirical distance between $\mathcal{E}(S_1)$ and $\mathcal{E}(S_2)$ is defined as:

$$\mathrm{d}_{\mathcal{E}}(S_1, S_2) = \frac{1}{2}\|\mathcal{E}(S_1) - \mathcal{E}(S_2)\|_1 = \frac{1}{2}\sum_{i=1}^m |p_i - q_i| \tag{8}$$

$\mathrm{d}_{\mathcal{E}}(S_1, S_2) \in [0, 1]$ compares the distance between two subsets through their competencies, rather than their real distributions. $\mathrm{d}_{\mathcal{E}}(S_1, S_2) = 0$ means that two subsets S_1 and S_2 are identical, while $\mathrm{d}_{\mathcal{E}}(S_1, S_2) = 1$ denotes S_1 and S_2 are completely different.

Example 7. In Fig. 3, given $S_1 = \{X_1, X_2, X_3\}$ and $S_2 = \{X_4, X_5, X_6\}$, $\mathcal{E}(S_1) = (\frac{5}{12}, \frac{1}{4}, \frac{1}{4}, \frac{1}{12})$ and $\mathcal{E}(S_2) = (\frac{1}{2}, \frac{1}{6}, 0, \frac{1}{3})$. $\mathrm{d}_{\mathcal{E}}(S_1, S_2) = \frac{1}{2}(|\frac{5}{12} - \frac{1}{2}| + |\frac{1}{4} - \frac{1}{6}| + |\frac{1}{4} - 0| + |\frac{1}{12} - \frac{1}{3}|) = \frac{1}{3}$.

To detect concept drift, we set $S_1 = w_p$, $S_2 = w_c$, and $\mathcal{X} = S_1 \cup S_2$. After $\mathrm{d}_{\mathcal{E}}(w_p, w_c)$ is computed, the two-sample non-parametric permutation test method [34] is applied to verify if $\mathrm{d}_{\mathcal{E}}(w_p, w_c)$ is significantly large (larger than the p-th percentile of a sample of $\mathrm{d}_{\mathcal{E}}(w_p, w_c)$ values obtained by the permutation test. Normally, p is pre-assigned as 5). Drift is identified between w_p and w_c once $\mathrm{d}_{\mathcal{E}}(w_p, w_c)$ is significantly large. Therefore, the competence-based empirical distance is a quantitative value that shows the degree of drift. If $\mathrm{d}_{\mathcal{E}}(w_p, w_c)$ is closer to 0, the degree of drift between w_p and w_c is low. While if $\mathrm{d}_{\mathcal{E}}(w_p, w_c)$ is closer to 1, the degree of drift between w_p and w_c is high.

B. Identify drift region data—CM-DRDI

The competence model-based drift region identification (CM-DRDI) is proposed to identify drift region data. CM-DRDI uses competence-based discrepancy density estimation (CDDE) [35] and competence-based empirical distance ($\mathrm{d}_{\mathcal{E}}$) as input. The main task of CM-DRDI is to find a threshold $\hat{\mathbb{F}}^*$ for CDDE. For any data instances X, X is drift region data if $\hat{\mathbb{F}}(X) \geq \hat{\mathbb{F}}^*$. The definition of $\hat{\mathbb{F}}(X)$ is defined as follows.

Definition 8. Competence Discrepancy Density Estimation (CDDE). Given $\mathcal{X} = w_p \cup w_c = \{X_1, \dots, X_n\}$, the subsets $S_1, S_2 \subseteq \mathcal{X}$, and $\Re(\mathcal{X}) = \{r_1, r_2, \dots, r_m\}$. For any $r_j \in \Re(\mathcal{X})$, the competence discrepancy density estimation [35] of given data instance X between w_p and w_c is defined as:

$$\begin{aligned} &\hat{\mathbb{F}}(X) = \frac{1}{m}\sum_{j=1}^m \mathbb{K}_j(X) \cdot \Omega(r_j), \\ &\mathbb{K}_j(X) = |H(r_j)|^{-d/2} \cdot K\left(H(r_j)^{-1/2} \cdot (P(r_j) - X)\right) \end{aligned} \tag{9}$$

where $K(\cdot)$ is the kernel function, $P(r_j) = \sum_{x \in r_j} x/|r_j|$ is the kernel point of r_j (Definition 8 in [35]), $\Omega(r_j)$ is the j^{th} element in the vector $|\mathcal{E}(w_p) - \mathcal{E}(w_c)|$ (discrepancy weight of r_j between w_p and w_c (Definition 7 in [35]), and $H(r_j)$ is the kernel bandwidth of r_j (Definition 9 in [35]).

Next, the uniform convergence bound of our $\hat{\mathbb{F}}(X)$ is given. Before that, we define the empirical distribution for any subsets $A \in \mathbb{R}^d$ as $\mathcal{F}_m = \frac{1}{m}\sum_{j=1}^m \mathbf{1}\{kp(r_j) \in A\}$. According to the [36], we have the following lemma.

Lemma 1. Define ellipsoid $B_{\mathbf{H}_0}(X, r) = \{X' \in \mathbb{R}^d : |\mathbf{H_0}^{-1/2}(X - X')| \leq r\}$, and $\mathcal{B} = \{B_{\mathbf{H}_0}(X, r) : X \in \mathbb{R}^d, r > 0,\ H_0 \text{ is a unit bandwidth}\}$. With probability at least $1 - 1/m$, the following holds uniformly for every $B \in \mathcal{B}$ and $\gamma \geq 0$:

$$\mathcal{F}(B) \geq \gamma \Rightarrow \mathcal{F}_m(B) \geq \gamma - \beta_m\sqrt{\gamma} - \beta_m^2, \tag{10}$$
$$\mathcal{F}(B) \leq \gamma \Rightarrow \mathcal{F}_m(B) \leq \gamma + \beta_m\sqrt{\gamma} + \beta_m^2, \tag{11}$$

where $\beta_m = 8d\sqrt{\log m/m}$.

Before giving the general upper and lower bounds for $\hat{\mathbb{F}}(X)$, we need to characterize how much the density can respectively decrease and increase from X in $B(X, r)$.

Definition 9. Given B defined in Lemma 1, we define

$$\begin{aligned} &\check{u}_x(r) = F(X) - \inf_{X' \in B(X,r)} F(X'), \\ &\check{u}_x(r) = \sup_{X' \in B(X,r)} F(X') - F(X). \end{aligned} \tag{12}$$

Theorem 1. Let v_d be the volume of the unit ball in $\mathbb{R}^d$. Then the following holds uniformly in $x \in \mathbf{R}^d$, $\epsilon > 0$, unit bandwidths $\mathbf{H}_0$, and $h > (\log m/m)^{1/d}$ with the probability at least $1 - 1/m$.

If $\int_{\mathbb{R}^d} K(u)\check{u}_x(h_{max}|u|/\sqrt{\sigma_d(\mathbf{H}_0)})du < \epsilon$, we have

$$\hat{\mathbb{F}}(X) \geq c_l \left(F(X) - \epsilon - C\sqrt{\frac{\log m}{m h_{max}^d}} \right), \tag{13}$$

and if $\int_{\mathbb{R}^d} K(u)\hat{u}_x(h_{max}|u|/\sqrt{\sigma_d(\mathbf{H}_0)})du < \epsilon$, we have

$$\hat{\mathbb{F}}(X) \leq c_u \left(F(X) + \epsilon + C\sqrt{\frac{\log m}{m h_{min}^d}} \right), \tag{14}$$

where $c_l = 1/(md_{max})$, $c_u = 1/(md_{min})$, $C = 8d\sqrt{v_d}||F||_\infty(\int_0^\infty k(t)t^{d/2}dt + 1) + 64d^2k(0)$, σ_d is the d^{th} eigenvalue of $\mathbf{H}_0$, $k(|u|) = K(u)$ (an assumption for $K(u)$), h_{min}, h_{max} are minimum and maximum values of h_j and d_{min}, d_{max} are minimum and maximum values of $\Omega(r_j), j = 1, ..., m$ $(H(r_j) = h_j\mathbf{H}_0)$.

The proof for Theorem 1 can be found in the Appendix.

Because h_{min}, h_{max}, d_{min} and d_{max} will decrease when m increasing, conditions of Theorem 1 can be satisfied if we have enough samples. This theorem shows that our estimation will approach the real distribution function of drift region at rate $\mathcal{O}((\log m/m)^{1/2})$, which provides a theoretical foundations for our following work. Next, the definitions to obtain the threshold of drift region are described as follows.

Definition 10. Drift Region Threshold. Given $\mathcal{X} = w_p \cup w_c = \{X_1, X_2, \dots, X_n\}$, drift region threshold is defined as:

$$\hat{\mathbb{F}}^* = \arg\min_{\hat{\mathbb{F}}(X_i)} \left(cdf(\hat{\mathbb{F}}(X_i)) \geq 1 - \mathrm{d}_\mathcal{E}(\mathcal{E}(w_p), \mathcal{E}(w_c)) \right), \tag{15}$$

$$cdf(\hat{\mathbb{F}}(X_i)) = \frac{|\{X_j : \hat{\mathbb{F}}(X_j) \leq \hat{\mathbb{F}}(X_i)\}|}{n}, j = 1, \dots, n, \tag{16}$$

where $\hat{\mathbb{F}}(X_i)$ is the CDDE of X_i, and $cdf(\hat{\mathbb{F}}(X_i))$ is the cumulative distribution function (CDF) of $\hat{\mathbb{F}}(X_i)$.

It should be noted that $\hat{\mathbb{F}}^*$ is not used to detect whether drift is presented unlike other papers, such as two-sample non-parametric permutation test [11], [23], [37]. The threshold is used to find the specific region of drift when a drift has occurred between two windows of data. Because $\hat{\mathbb{F}}(X)$ will approach the real distribution function when m increases, it is clear that $cdf(\hat{\mathbb{F}}(X))$ will approach the real one when m increases. Fig. 4 shows an example of obtaining $\hat{\mathbb{F}}^*$. The x-axis is the CDDE value, and the y-axis is the CDF of the CDDE value. The data instance which has a larger CDDE has a higher chance of being located in the drift region. Since the competence-based empirical distance reveals the relative difference between w_p and w_c, CM-DRDI utilizes the competence-based empirical distance to determine how many data instances should be marked as drift region data. Therefore, the minimum CDDE value whose CDF is equal to $1 - \mathrm{d}_\mathcal{E}$ is used as the threshold.

By applying the outputs of CM-DD with kernel density estimation, CDDE accurately maps the drift-affected discrepancy from one-dimensional competence measurement space to the multi-dimensional data feature space. However, the result of CDDE is highly affected by noisy data. To solve this issue, the Recursion Noise Reduction (RNR) algorithm is applied on

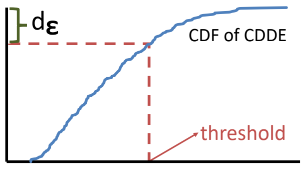


Fig. 4: An example of obtaining $\hat{\mathbb{F}}^*$.

Algorithm 1 Recursion Noise Reduction (RNR)

```
Require:
    A set of data $\mathcal{X} = \{X_1, X_2, \dots, X_n\}$
 1: $CheckSet \leftarrow \varnothing$
 2: for all $X_i \in \mathcal{X}$ do
 3:     Get $\mathcal{P}(X_i)$ and $\mathcal{L}(X_i)$
 4:     if $|\mathcal{L}(X_i)| > 0$ then
 5:         $CheckSet \leftarrow CheckSet \cup \{X_i\}$
 6:     end if
 7: end for
 8: $CheckList \leftarrow$ sorted $X_i \in CheckSet$ by $|\mathcal{L}(X_i)| \downarrow$
 9: while $CheckList \neq \varnothing$ do
10:     $X \leftarrow$ the first item of $CheckList$
11:     $MisclassifiedFlag \leftarrow false$
12:     for all $X_i \in \mathcal{P}(X)$ do
13:         if $X_i$ cannot be correctly classified by $\mathcal{X} - \{X_i\} - \{X\}$ then
14:             $MisclassifiedFlag \leftarrow true$; break
15:         end if
16:     end for
17:     $CheckSet \leftarrow CheckSet - \{X\}$
18:     if $MisclassifiedFlag == false$ then
19:         $\mathcal{X} \leftarrow \mathcal{X} - \{X\}$
20:         for all $X_i \in CheckSet(X)$ do
21:             if $X \in \mathcal{L}(X_i)$ then
22:                 $\mathcal{L}(X_i) \leftarrow \mathcal{L}(X_i) - \{X\}$
23:                 if $|\mathcal{L}(X_i)| == 0$ then
24:                     $CheckSet \leftarrow CheckSet - \{X_i\}$ ; continue
25:                 end if
26:             end if
27:             if $X \in \mathcal{P}(X_i)$ then
28:                 $\mathcal{P}(X_i) \leftarrow \mathcal{P}(X_i) - \{X\}$
29:             end if
30:         end for
31:         $CheckList \leftarrow$ sorted $X_i \in CheckSet$ by $|\mathcal{L}(X_i)| \downarrow$
32:     end if
33: end while
34: return $\mathcal{X}$
```

each time window to remove noisy data and preserve potential new concept data.

Definition 11. Positive Set. For any $X_i \in \mathcal{X}$, the positive set of X_i is defined as:

$$\begin{aligned} \mathcal{P}(X_i) = \{X_j \in \mathcal{X} : X_i \in kNN(X_j), y_j == y_i, \\ X_j \text{ is correctly classified by } kNN(X_j)\} \end{aligned} \tag{17}$$

where $kNN(X_j)$ is the k-nearest neighbors of X_j, y is the class label of X.

Definition 12. Liability Set. For any $X_i \in \mathcal{X}$, the liability set of X_i is defined as:

$$\mathcal{L}(X_i) = \{X_j \in \mathcal{X} : X_i \in kNN(X_j), y_j \neq y_i\} \tag{18}$$

Algorithm 1 lists the procedure of RNR. At line 3, the first step is to get the PositiveSet (Definition 11) and LiabilitySet (Definition 12) of each data instance. Any data instance whose LiabilitySet is not null is added to $CheckSet$ (line 4–6). At line 8, the $CheckSet$ is ordered in an descending order of the size of LiabilitySet. From line 9 to line 33, the recursion loop removes the data which are confirmed as noisy data from the data sample. If any data instance X_i in $\mathcal{P}(X)$ cannot be correctly classified by $\mathcal{X} - \{X_i\} - \{X\}$, then X is not noisy data. If X is confirmed as noisy data, X is removed (line 19). Then, the PositiveSet and LiabilitySet of each data instance in $CheckSet$ is updated (line 20–20). At line 24, the data instance whose LiabilitySet is null is removed from

Algorithm 2 CM-DRDI

```
Require:
    Two data time windows $w_p$ and $w_c$ where there is a concept drift occurred.
    Preparation Stage:
1: $w_p \leftarrow \mathrm{RecursionNoiseReduction}(w_p)$
2: $w_c \leftarrow \mathrm{RecursionNoiseReduction}(w_c)$
3: Get $\mathrm{d}_{\mathcal{E}}(w_p, w_c)$
4: for all $X_i \in \mathcal{X} = w_p \cup w_c$ do
5:     Get $\hat{\mathbb{F}}(X_i)$ through Definition 8
6: end for
7: Get $\hat{\mathbb{F}}^*$ through Definition 10
    Evaluation Stage:
8: For any data instance $X$, get $\hat{\mathbb{F}}(X)$ through Definition 8
9: if $\hat{\mathbb{F}}(X) \geq \hat{\mathbb{F}}^*$ then
10:     return $true$                          ▷ $X$ is drift region data
11: else
12:     return $false$                         ▷ $X$ is not drift region data
13: end if
```

$CheckSet$. $CheckSet$ is reordered (line 31) and the recursion loop is repeated until $CheckSet$ is null.

In summary, the whole procedure of CM-DRDI is listed in Algorithm 2. CM-DRDI has two stages: 1) preparation stage: to calculate $\hat{\mathbb{F}}^*$; and 2) evaluating stage to evaluate if a data instance X is in the drift region. After CM-DD has confirmed that concept drift has occurred between w_p and w_c, CM-DRDI applies RNR to remove noisy data (line 1–1). Then, CM-DRDI recalculates the competence-based empirical distance $\mathrm{d}_{\mathcal{E}}(w_p, w_c)$ (line 3) and the CDDE values of each data instance (line 4–6). In the last step of the preparation stage, CM-DRDI obtains the threshold of CDDE $\hat{\mathbb{F}}^*$ through Definition 10 (line 7). In the evaluation stage, CM-DRDI first calculates the CDDE values of a data instance X (line 8). If $\hat{\mathbb{F}}(X)$ is greater than $\hat{\mathbb{F}}^*$, this means that X is a data instance located in the drift region.

C. Filtering drift region instances—CM-DRIFT

Competence model-based drift region filtering(CM-DRIFT) is to filter the data sample used for learning, which is a concept drift adaptation method. CM-DRIFT relies on two pieces of input information: 1) the output of CM-DD, which indicates whether a concept drift has occurred; 2) the output of CM-DRDI, which evaluates if a data instance is located in the drift region. CM-DRIFT adopts an intuitive idea that: 1) when concept drift occurs, it removes the obsolete drift region data from the data sample, and adds drift region data which belongs to the time window w_c, into the data sample; 2) whether concept drift occurs or not, it uses the prediction history of data instances to enable/disable the data instance to enhance the ability of the data sample to represent the current concept. The details of CM-DRIFT are listed in the pseudo-code of Algorithm 3.

As shown in Fig. 2, the procedure of CM-DRIFT is implemented immediately after CM-DD or CM-DRDI. The input of CM-DRIFT comprises four parts: 1) $\mathcal{X}$ represents the current data sample; 2) Boolean variable $DriftFlag$ shows whether concept drift has occurred; 3) w_p and w_c are used to identify drift region data; 4) X_t is the last data instance with a class label.

The first step of CM-DRIFT is to remove noisy data from w_c by RNR. The cleaned data are stored in $W_{\text{new}'}$ (line 1). When concept drift occurs, all data instances in the current data sample $\mathcal{X}$ need to be checked by CM-DRDI to evaluate if they are located in the drift region (line 3–8). All drift

Algorithm 3 CM-DRIFT

```
Require:
    The data sample $\mathcal{X}$.
    The output of CM-DD $DriftFlag$, which indicates the status of drift.
    Two data time windows $w_p$ and $w_c$.
    New labeled data instance $X_t$.
1: $W_{\text{new}'} \leftarrow \mathrm{RecursionNoiseReduction}(w_c)$
2: if $DriftFlag == true$ then
3:     $DriftSet \leftarrow \varnothing$
4:     for all $X_i \in \mathcal{X}$ do
5:         if CM-DRDI$(w_p, w_c, X_i) == true$ then
6:             $DriftSet \leftarrow DriftSet \cup \{X_i\}$
7:         end if
8:     end for
9:     for all $X_i \in DriftSet$ do
10:        Get $kNN(X_i)$ in context of $DriftSet \cup W_{\text{new}'}$
11:        for all $X_j \in kNN(X_i)$ and $X_j \in W_{\text{new}'}$ do
12:            if $ClassLabel(X_i) \neq ClassLabel(X_j)$ then
13:                $\mathcal{X} \leftarrow \mathcal{X} - \{X_i\}$; continue
14:            end if
15:        end for
16:    end for
17:    for all $X_i \in W_{\text{new}'}$ do
18:        if CM-DRDI$(w_p, w_c, X_i) == true$ then
19:            $\mathcal{X} \leftarrow \mathcal{X} \cup \{X_i\}$
20:        end if
21:    end for
22: else
23:    if $X_t \in W_{\text{new}'}$ then
24:        $\mathcal{X} \leftarrow \mathcal{X} \cup \{X_t\}$
25:        Get $kNN(X_t)$ in context of $\mathcal{X}$
26:        for all $X_i \in kNN(X_t)$ do
27:            if $ClassLabel(X_i) \neq ClassLabel(X_t)$ then
28:                $\mathcal{X} \leftarrow \mathcal{X} - \{X_i\}$
29:            end if
30:        end for
31:    end if
32: end if
33: for all $X_i \in \mathcal{X}$ do
34:    Enable/Disable $X_i$ based on its prediction accuracy $p_i$
35: end for
```

region data identified by CM-DRDI are stored in $DriftSet$ (line 6). CM-DRDI may misidentify some data instances that are actually located in the non-drift region, and these data may be useful for prediction. To solve this problem, we only remove the drift region data that is in conflict with the new concept (line 9–16). To implement this solution, for each data instance X_i in $DriftSet$, we find its kNN in the context of $DriftSet \cup W_{\text{new}'}$ (line 10). If any X_j in $kNN(X_i)$ belongs to the new concept, and X_i and X_j have different class labels, X_i is considered to have conflicts with the new concept and will be removed from data sample $\mathcal{X}$ (line 13). After this, in the cleaned time window $W_{\text{new}'}$, data instances marked as drift region data will be added to data sample $\mathcal{X}$ (line 17–21).

If there is no concept drift, the first step is to verify if the new data X_t is noisy (line 23). If new data X_t is not removed by RNR, X_t will be added into the data sample $\mathcal{X}$. For each data instance X_i in $kNN(X_t)$, it will be checked if it is in conflict with the new data X_t. If X_i and X_t have different class labels, X_i will be removed. Because X_t brings the new concept into the data sample, removing the conflicting data instance will help the data sample to keep the new concept as much as possible.

To ensure the data sample represents the new concept, we introduce the prediction history of X_i to control the effectiveness of the data instance. The prediction history is denoted by the array A of maximum length l. which records the latest prediction accuracy of X_i when X_i is a retrievable data instance. A retrievable data instance means the $\mathrm{dist}(X_i, X_t) \leq \mathrm{dist}(X_k, X_t)$, where $\mathrm{dist}(X_i, X_t)$ is the distance between X_i and X_t, and X_k is the kth nearest neighbor of X_t. If X_i and X_t have the same class label, 1 will be pushed into A; otherwise, 0 will be pushed into A. If array A reaches the maximum length l, the last item of A will be dropped. Therefore, we can obtain the prediction accuracy p_i of data instance X_i by (19), where l_A is the length of array A.

$$p_i = \sum_{a \in A} a/l_A \tag{19}$$

CM-DRIFT adopts the same confidence interval test used in the IB3 algorithm [38] and the PECS algorithm [39] to control the effectiveness of the data instance. Then, the confidence interval of the prediction accuracy of X_i is calculated below.

$$\text{confidence interval} = \frac{p_i + \frac{z^2}{2l_A} \pm z\sqrt{p_i(1-p_i)/l_A + z^2/4l_A^2}}{1 + z^2/l_A} \tag{20}$$

where z is the confidence interval coefficient, which is either tabulated or computed. If the upper bound of the confidence interval falls below the inactivation threshold p_{max}, X_i is temporarily disabled from the data sample for training the learner; on the contrary, X_i may eventually be re-enabled to the data sample if its lower bound rises above the agreement acceptance probability p_{min}.

IV. Experimental evaluation

CM-DRIFT is a drift adaptation method based on the drift understanding process. It can be combined with any detection method. Conventional detection-based adaptation methods retrain/tune the outdated model when drift is detected. However, they do not consider that the data instances used to retrain/tune the model could present the old pattern. CM-DRIFT is proposed to solve that problem. Therefore, the experimental evaluation in this paper mainly focuses on two aspects: 1) whether the drift region is successfully identified; and 2) how the identified drift region improves the classification accuracy?

The experimental evaluation consists of three sections with ten experiments. Section IV-A validates the effectiveness of CM-DRDI by two synthetic data. Section IV-B compares CM-DRIFT to seven concept drift adaptation methods on four synthetic data and two real-world data. Section IV-C evaluates the performance of CM-DRIFT with different parameters, and Section IV-D evaluates CM-DRIFT with different base learners. Section IV-E give the computation efficiency of the proposed method.

Data. Six datasets of concept drift are used in this section: SEA [27], Rotating Hyperplane (RH) [13], Usenet1 and Usenet2 [40], [41], NOAA weather [28] and Spam [40], [42]. These data have been widely used in the research of concept drift and detailed description of these data can be found in the cited papers.

Measurement. In this paper, we assume that the instances are obtained one-by-one in temporal order to mimic a real data stream scenario. CM-DRIFT is proposed to respond to drift accurately and quickly when detected. Therefore, our goal is to improve the online classification performance. In data stream mining, the prequential evaluations are mainly used as criteria whereby each instance is first used to test the model, and then to retrain/tune the model. We use prequential accuracy, prediction, recall, and F1 score as the evaluations for the classification performance on data streams [43]. In addition, to show the capacity of drift identification in our method, we plot and compare the identified drift regions identified by our method with two baseline methods where we manually add drift with different degrees in the synthetic data. Furthermore, we use the statistical test to validate the significance of the classification performance results.

A. Evaluating CM-DRDI

SEA and RH are used to validate how many drift region data can be correctly retrieved by CM-DRDI. In this section, CM-DRDI is compared to TPCA and MTPCA method. TPCA [21] can also identify drift region data. However, TPCA uses one threshold of 0.05 for all datasets. To conduct comprehensive comparisons, we modify the threshold in TPCA to be equal to $\mathrm{dist}_{\mathrm{CM}}$ so that TPCA can suit different datasets. The modified TPCA is denoted by MTPCA.

For each synthetic dataset, several tests with different concept drift parameter settings are conducted. In each test, there are data from two concepts. The *first concept* represents the previous data pattern before drift has occurred, and the *second concept* represents the current data pattern. As the data is synthetically generated, the concept drift region is known in advance. The task of each test is to identify the data instances belonging to the second concept that are located in the drift region.

Experiment 1. The SEA dataset. SEA contains three features x_1, x_2, x_3, and the class decision boundary is given by $x_1 + x_2 = \theta$. Sudden drift occurs when the value of θ changes. In addition, noise is introduced by randomly switching the labels of 10% of the data instances. There are four tests in this experiment. Each test in this experiment involves two concepts, and each concept has 500 data instances. In all tests, the θ value of the first concept is 7, and θ values of the second concept is 8, 8.5, 9.5, 11 for **Test 1-4** separately. To display the result of drift region identification in a 2D-figure, we remove the irrelevant attribute x_3. Fig. 5 shows test results of drift region identification on the SEA dataset. In Fig. 5, the *sub-figures in the same column* are the results of the same test. In each column of sub-figures, the 1st and 2nd sub-figures present the data instances belonging to the first concept and second concept separately; the 3rd sub-figure draws the drift region and the 4th-6th sub-figures draw drift region identified by CM-DRDI, TCPA, and MTPCA separately.

As shown in Fig. 5, as the degree of drift becomes larger (larger θ for the second concept), drift region contains more data instances. CM-DRDI identifies almost all the drift region data, and picks out a few non-drift region data located very close to the drift region. TCPA only identifies some drift region data, and also a few data that are located outside the drift region. Compared to TCPA, MTPCA can identify more drift region data; however, more data instances located outside the drift region are also identified.

Fig. 6 shows results of recall, precision, and F1 score of these three methods. The recall values of CM-DRDI in the four tests are very close to 1, which means CM-DRDI well identifies drift region. The precision values of CM-DRDI increase as θ becomes large, which means fewer non-drift region data will be identified when the degree of drift is larger.

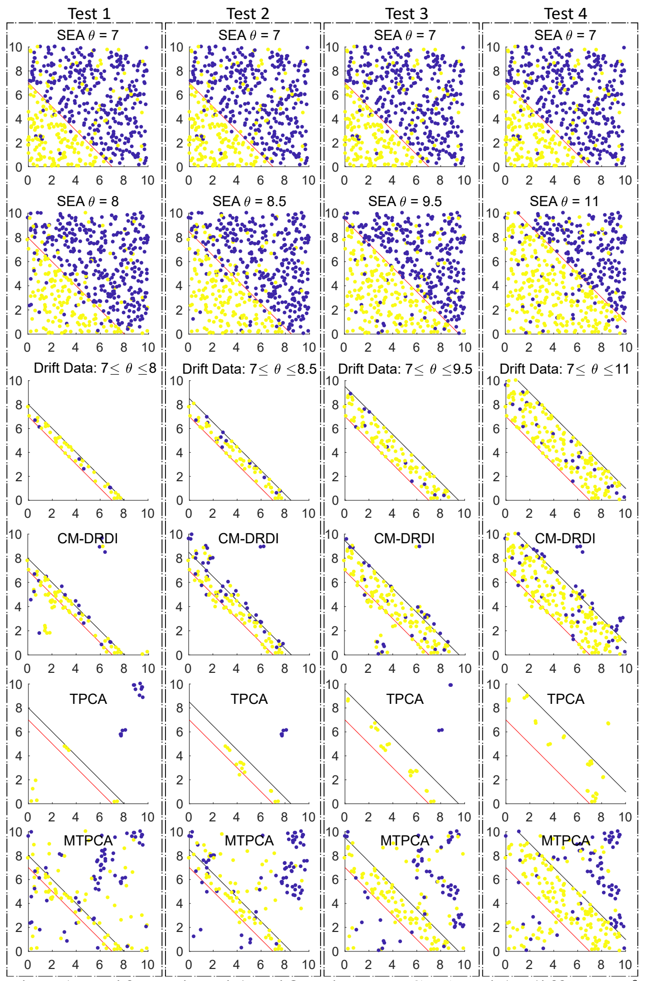


Fig. 5: Drift region identification on SEA with different θ.

The F1 scores of CM-DRDI show that it outperforms TPCA and MTPCA. The recall values of TPCA are only around 0.2, which indicates that only 20% of drift region data have been identified. Even though the precision values of TPCA are higher than the other two methods, the F1 scores of TPCA are relative small. MTPCA has good performance in recall but not in precision, which means that MTPCA does not improve TPCA's performance in identifying drift region.

Discussion. The Recall subplot in Fig. 6 denotes how many instances from the true drift region are successfully identified by CM-DRIFT. The Precision subplot in Fig. 6 denotes how many instances identified as drift region by CM-DRIFT are truly in the drift region (i.e., $1-$false positive rate). Although CM-DRIFT sometimes wrongly identifies many instances as drift region (the precision is not always high), it can successfully identify most true drift region instances (the recall is high). For example, when θ is 1, recall is 98% and precision is 53%, which means that 47% identified instances are wrong. Fig. 5 gives a deep view of these wrongly identified instances. According to Fig. 5, we found that these wrongly identified instances are closed to the true drift region. Therefore, the classification accuracy is not significantly affected by these wrongly identified instances.

Experiment 2. The Rotating Hyperplane dataset (RH).

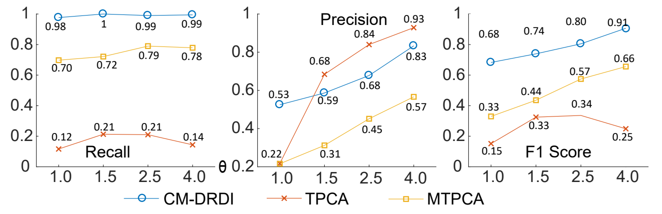


Fig. 6: Drift region identification on SEA with different θ.

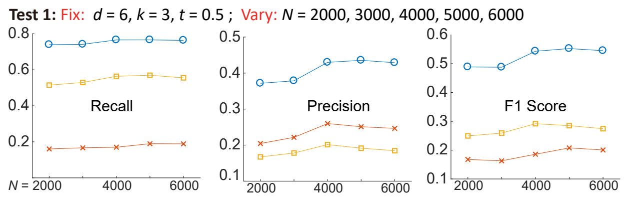
The parameter settings of the gradual drift in RH have been described in [44]. In summary, three parameters (N, k and t) control the gradual drift, and the parameter d controls the data dimension. The concept changes gradually during N data instances. Noise is introduced by randomly switching the label of 10% of the data instances.

Four tests are conducted to validate the effectiveness of drift region identification. In Test 1-3, we only change one of the parameters and fix the others. In Test 4, k and d are changed simultaneously. Fig. 7a–7d show the results of these experiments. In Fig. 7a and Fig. 7b, we find that the precision of each drift region identification method improves a little when the size of the concept increases or the number of changing dimensions increases. From Fig. 7c, we find that the performance of each method improves when the degree of drift is large, since the magnitude of the change defines the degree of drift. Fig. 7d shows that the performance of each method decreases as the dimension of data increases. As the data feature space grows exponentially as the data dimension increases linearly, it is more challenging to identify drift region data. CM-DRDI can still identify over 60% of drift region data, which is much better than TPCA and MTPCA. From the results of all 4 test groups, we conclude that CM-DRDI outperforms TPCA and MTPCA in precision, recall and F1 score.

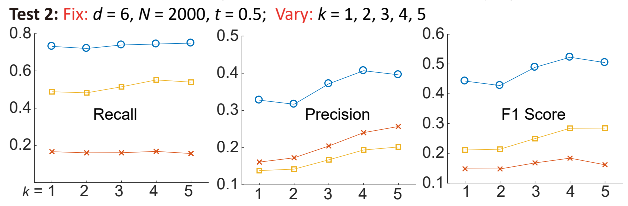
B. *Evaluating CM-DRIFT*

In this subsection, CM-DRIFT is compared to seven drift adaptation methods. The selected baselines are: 1) DDM [10] retrains learners by an online error rate-based drift detection method; 2) HDDM(A-test) [45] retrains learners by the Hoeffding's inequality-based drift detection method with A-test; 3) HDDM(W-test) [45] is similar to HDDM(A-test) but uses W-test; 4) ECDD [46] retrains learner by the EWMA chart-based drift detection method; 5) NEFCS [21] is a kNN-based adaptive model that uses the output of the competence model-based drift detection method to discard the data instances affected by concept drift. 6) NSE [28] is an adaptive ensemble that uses dynamic voting weights to handle concept drift; 7) SAMkNN [20] is an adaptive ensemble that keeps past concepts in memory and considers all buffered concepts to make a final prediction. NEFCS is implemented in MATLAB with default parameters. All other baseline methods were implemented with default parameters via MOA [47].

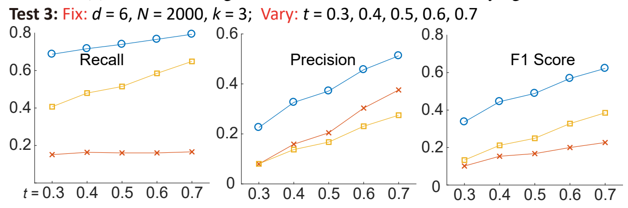
In CM-DRIFT, we use kNN as the base learner with $k = 5$. We use $k = 5$ because this is the default value of most kNN-based concept drift adaptation methods. In all baseline methods, the value of k is also 5, which is their default value. The size of the time window is equal to the size of the training set. The distance function used in CM-DRIFT is Euclidean.



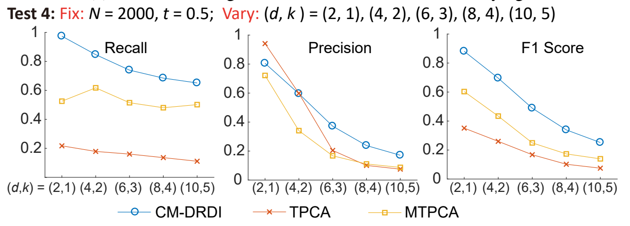
(a) Test 1: drift region identification on RH with varying N.



(b) Test 2: drift region identification on RH with varying k.

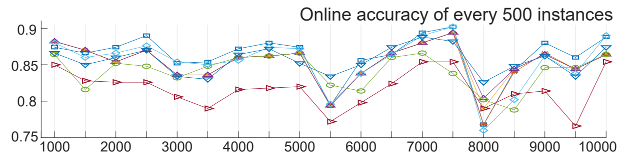


(c) Test 3: drift region identification on RH with varying t.

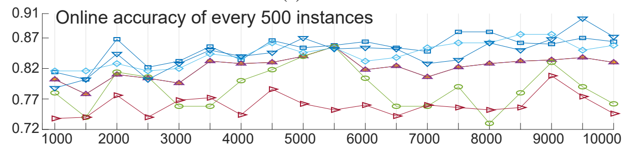


(d) Test 4: drift region identification on RH with varying d and k.

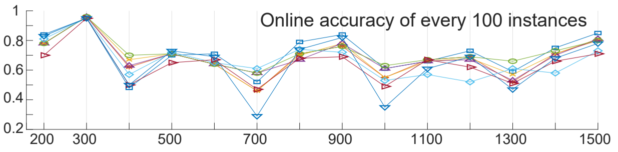
Fig. 7: Four tests in Experiment 2.



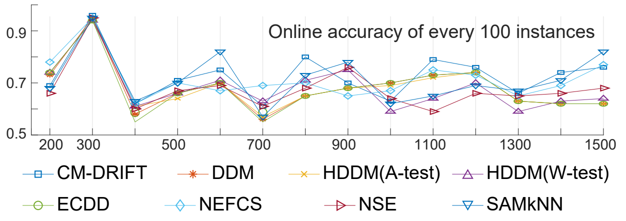
(a) SEA.



(b) RH.



(c) Usenet1.



(d) Usenet2.

Fig. 8: The online classification accuracy on different datasets.

Other parameters required by CM-DRIFT are l_A, $p_{\min}$, $p_{\max}$, and z. We adopted the same values suggested by the authors in [21], which are $l_A = 10$, $p_{\min} = 0.5$, $p_{\max} = 0.5$. For the parameter z, we set $z = 1.96$, which indicates the critical value for a 95% confidence interval. The 95% confidence interval is a common value used in statistics.

We evaluate CM-DRIFT on four synthetic concept drift datasets and two real-world datasets. The results on the synthetic datasets demonstrate CM-DRIFT's performance on different types of concept drift. The results on the real-world datasets demonstrate CM-DRIFT's performance in real-world situations.

Experiment 3. The SEA dataset. The experiment is set up as follows. Each concept block contains $2,500$ random data instances, and there are a total of $10,000$ data instances. The first 500 data instances from the first block are used as the initial training set. The remaining $9,500$ instances are classified and learned incrementally.

Fig. 8a shows the online classification accuracy of baselines on the SEA dataset. The online classification accuracy at each time point is reported based on the most recent 500 data instances. According to this experiment's setting, three sudden drifts occur at time points $2,500$, $5,000$, and $7,500$. CM-DRIFT increased in accuracy as the number of instances increased while there is no drift. It also had the highest online accuracy before drift occurs and, immediately after drift presents, the accuracy of all baselines dropped – for example, at time points, $3,000$, $5,500$, and $8,000$. However, CM-DRIFT was the quickest to respond drift and has the highest accuracy at next report time point ($3,500$, $6,000$, and $8,500$). Once the concept stabilizes, CM-DRIFT's performance is consistently at the forefront. Plus, CM-DRIFT had the highest overall accuracy over the whole period, as shown in TABLE I.

Experiment 4. The Rotating Hyperplane dataset (RH). The experiment is set up as follows. The dataset has a total of $10,000$ data instances with $d = 10$ dimensions. The rest of the parameters are set up as $K = 5$, $T = 0.5$, and $N = 1000$. Concept drift occurs gradually over all $10,000$ instances. The first 500 data instances are used as the training set. The remaining $9,500$ instances are classified and learned incrementally. Fig. 8b shows the online classification accuracy of baselines on the RH dataset. The online classification accuracy at each time point is reported based on the most recent 500 data instances. Since the concept gradually drifts in the RH dataset all the time, it is more challenging to track concepts continually. Over all 19 reported time points, CM-DRIFT has the highest accuracy for 10 times, second highest for 7 times, and third highest for twice. The average online accuracy rank for CM-DRIFT is 1.57. According to the results in TABLE I, CM-DRIFT has the highest values in both Accuracy and F1 Score, which indicates that our algorithm is superior to the other algorithms in dealing with gradual drift.

Experiment 5. The Usenet datasets. Each data instance in Usenet1 and Usenet2 is derived from Usenet posts in the 20 Newsgroups collection . Each dataset consists of $1,500$ data instances. Each data instance has 99 attributes. The task is to classify messages as either interesting or junk as they

TABLE I: Comparison among learning algorithms.

Datasets	Methods	Accuracy (rank)	Precision	Recall	F1 Score (rank)	Time
SEA	CM-DRIFT	0.8623 (1)	0.8598	0.9229	0.8903 (1)	27.63
	DDM	0.8521 (6)	0.8548	0.9102	0.8816 (6)	1.97
	HDDM(A-Test)	0.8534 (5)	0.8614	0.9029	0.8817 (5)	1.79
	HDDM(W-Test)	0.8541 (4)	0.8617	0.9040	0.8823 (4)	1.73
	ECDD	0.8418 (7)	0.8539	0.8909	0.8720 (7)	1.31
	NEFCS	0.8542 (3)	0.8557	0.9130	0.8834 (3)	23.94
	NSE	0.8166 (8)	0.8283	0.8793	0.8530 (8)	100.00
	SAMkNN	0.8563 (2)	0.8589	0.9125	0.8849 (2)	1.00
RH	CM-DRIFT	0.8520 (1)	0.8513	0.8580	0.8547 (1)	30.94
	DDM	0.8215 (4)	0.8258	0.8213	0.8235 (4)	4.71
	HDDM(A-Test)	0.8215 (4)	0.8258	0.8213	0.8235 (4)	4.48
	HDDM(W-Test)	0.8215 (4)	0.8258	0.8213	0.8235 (4)	4.42
	ECDD	0.7880 (7)	0.7933	0.7871	0.7902 (7)	2.43
	NEFCS	0.8446 (2)	0.8445	0.8502	0.8473 (2)	23.69
	NSE	0.7596 (8)	0.7641	0.7609	0.7625 (8)	22.20
	SAMkNN	0.8444 (3)	0.8453	0.8485	0.8469 (3)	1.42
Usenet1	CM-DRIFT	0.7157 (1)	0.7376	0.7094	0.7232 (1)	3.82
	DDM	0.6843 (5)	0.7181	0.6535	0.6843 (6)	1.68
	HDDM(A-Test)	0.6914 (4)	0.7197	0.6726	0.6954 (5)	1.13
	HDDM(W-Test)	0.6950 (3)	0.7198	0.6835	0.7012 (3)	1.13
	ECDD	0.7157 (1)	0.7417	0.7012	0.7209 (2)	1.24
	NEFCS	0.6650 (6)	0.6618	0.7367	0.6972 (4)	2.92
	NSE	0.6407 (8)	0.6474	0.6890	0.6676 (7)	4.84
	SAMkNN	0.6521 (7)	0.6940	0.6003	0.6438 (8)	0.37
Usenet2	CM-DRIFT	0.7271 (1)	0.7453	0.8971	0.8142 (1)	3.62
	DDM	0.6821 (4)	0.7037	0.9035	0.7912 (7)	2.73
	HDDM(A-Test)	0.6821 (4)	0.6990	0.9185	0.7939 (5)	2.95
	HDDM(W-Test)	0.6814 (6)	0.6946	0.9314	0.7958 (4)	2.24
	ECDD	0.6793 (7)	0.7061	0.8885	0.7869 (8)	2.96
	NEFCS	0.7157 (2)	0.7526	0.8542	0.8002 (3)	3.53
	NSE	0.6786 (8)	0.6962	0.9185	0.7921 (6)	3.22
	SAMkNN	0.7157 (2)	0.7378	0.8896	0.8066 (2)	0.37
NOAA Weather	CM-DRIFT	0.7906 (1)	0.8098	0.9086	0.8564 (1)	67.34
	DDM	0.7634 (4)	0.6389	0.5612	0.5975 (3)	6.33
	HDDM(A-Test)	0.7626 (5)	0.6417	0.5469	0.5905 (4)	5.93
	HDDM(W-Test)	0.7516 (6)	0.6221	0.5254	0.5697 (6)	3.62
	ECDD	0.7423 (7)	0.6006	0.5277	0.5618 (7)	1.94
	NEFCS	0.7839 (2)	0.8230	0.8732	0.8474 (2)	54.45
	NSE	0.7054 (8)	0.5360	0.4550	0.4922 (8)	47.09
	SAMkNN	0.7777 (3)	0.7122	0.4891	0.5800 (5)	2.05
Spam	CM-DRIFT	0.9507 (3)	0.9184	0.8723	0.8948 (3)	151.03
	DDM	0.9216 (5)	0.8910	0.7679	0.8249 (5)	161.55
	HDDM(A-Test)	0.9190 (6)	0.8728	0.7761	0.8216 (6)	111.42
	HDDM(W-Test)	0.9290 (4)	0.9055	0.7866	0.8419 (4)	210.00
	ECDD	0.9027 (7)	0.8304	0.7478	0.7870 (7)	63.37
	NEFCS	0.9581 (1)	0.9190	0.9056	0.9123 (1)	146.09
	NSE	0.6084 (8)	0.3253	0.5860	0.4184 (8)	1577.35
	SAMkNN	0.9567 (2)	0.9531	0.8623	0.9054 (2)	24.97

TABLE II: The average rank of accuracy and F1 score for learning algorithms.

Algorithms	Average rank on accuracy	Average rank on F1 score	Average computation time
CM-DRIFT	**1.33**	**1.33**	47.40
DDM	4.67	5.17	29.83
HDDM(A-Test)	4.67	4.83	21.28
HDDM(W-Test)	4.50	4.17	37.19
ECDD	6.00	6.33	12.21
NEFCS	2.67	2.50	42.44
NSE	8.00	7.50	292.45
SAMkNN	3.17	3.67	**5.03**

arrive. Usenet1 simulates sudden drift scenarios and Usenet2 simulates reoccurring drift [41].

Fig. 8c and Fig. 8d show the online classification accuracy of baselines on Usenet1 and Usenet2. The online classification accuracy at each time point is reported based on the most recent 100 data instances. In Fig. 8c, CM-DRIFT consistently responded quickly to sudden drift and has the highest accuracy prior to the next drift occurring, for example, the online accuracy at time points 600, 900 and $1,200$. In Fig. 8d, CM-DRIFT maintains a part of the data sample to deal with possible drift recurrences in the future. As a result, the accuracy of CM-DRIFT suffers slightly in comparison to SAMkNN. Nevertheless, CM-DRIFT can deal with the situation of concept reoccurring (e.g., 301–600 and 901–$1,200$) with the highest accuracy at time point $1,200$.

Experiment 6. The NOAA weather dataset. TABLE I lists the overall classification accuracy of baselines on the NOAA weather dataset. Even though most of the learning algorithms have similar overall accuracy, the F1 scores of these methods are less than 60%, which means the accuracy of predicting rain is less than 60%. CM-DRIFT outperforms the other methods not only in overall accuracy but also in the F1 score of the minority class, which means CM-DRIFT can handle concept drift in an imbalanced data environment.

Experiment 7. The spam dataset. TABLE I lists the overall classification accuracy of baselines on the spam dataset. Although the overall accuracy of CM-DRIFT is only the third highest, its accuracy is very closed to the best and the second best methods.

TABLE I lists the performance of all methods on different datasets. In addition to the evaluation measurements, this table summarizes the ranking of accuracy, the ranking of F1 score, and the computation time in seconds. TABLE II lists the average performance of all methods on six datasets. We use Friedman-post-hoc test after Conover to test whether the difference of accuracy between CM-DRIFT and other baseline methods is significant [12]. The results are shown in TABLE III where $|R_i - R_{\text{CM-DRIFT}}|$ is the value of the rank difference between i-th baseline and CM-DRIFT over all data streams. For example, 18 is the rank difference between DDM and CM-DRIFT. The p-value row is the corresponding p-value for $|R_i - R_{\text{CM-DRIFT}}|$. The analysis results of these experiments can be summarized as follows:

1) Generally, TABLE III shows that our method is significantly better than other baseline methods at the significance level of 0.05. The proposed CM-DRIFT outperforms the other learning algorithms on five datasets and for the spam email datasets, the performance of CM-DRIFT is very close to the best performing learning algorithm. CM-DRIFT and NEFCS, both being adaptive models, have better performance than the retraining models, which indicates that adaptive models are more suitable for handling concept drift on the tested data.

2) The online classification accuracy of CM-DRIFT on the synthetic datasets shows that CM-DRIFT reduces the recovery time of accuracy when concept drift occurs. The strategy of filtering drift region data is effective and CM-DRIFT can adapt to different types of concept drift.

3) Regarding computational complexity, it can be seen that CM-DRIFT and NEFCS require more computation time than the retraining models. This is because the data distribution-based drift detection method used in these methods incurs more computation cost than the learner error-based drift detection method. Nevertheless, CM-DRIFT shows no manifest disadvantages over other baselines. NSE has the highest computation time since it needs more computation cost to evaluate and weight the multiple base learners.

TABLE III: Statistical test based on the classification accuracy in Table I

Post-hoc test after Conover	DDM	HDDM (A-Test)	HDDM (W-Test)	ECDD	NEFCS	NSE	SAMkNN
$\lvert R_i - R_{\text{CM-DRIFT}} \rvert$	18	18	16	24	7	35	10
p-value	4E-10	4E-10	6E-09	3E-13	1E-3	4E-18	2E-05

C. Evaluating CM-DRIFT with different parameters

The CM-DRIFT algorithm utilizes two time windows and the competence model to filter drift region data. The size of the time window defines how many data instances are used to represent one concept. If the size is too small, the

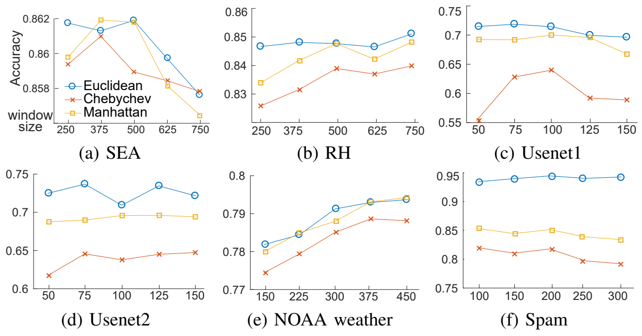


Fig. 9: The classification accuracy of CM-DRIFT with different window sizes and different distance functions.

data instances in the latter time window cannot correctly represent the new concept. If the size is too large, the latter time window may cover the previous concept leading to bad performance. The distance function defines the Solve rule for constructing the competence model and the condition if any two data instances could be k-nearest neighbors. Therefore, the size of the time window and the distance function have an impact on the performance of CM-DRIFT. In this section, we describe two experiments to evaluate the performance of CM-DRIFT with different sized time windows and different distance functions. The remaining parameters are set to the default values.

Experiment 8. Varying the size of the time window. To evaluate how the size of the time window affects the performance of CM-DRIFT, we set the size of the time window to $0.5\times$, $0.75\times$, $1\times$, $1.25\times$, or $1.5\times$ of the size number set in the Section IV-B.

Experiment 9. Varying the distance function. To evaluate how the distance function affects the performance of CM-DRIFT, in addition to Euclidean distance, two of the most commonly used distance function, Manhattan distance and Chebyshev distance, are used in this experiment.

Fig. 9 plots the classification accuracy of CM-DRIFT with different parameters on the six datasets. From Fig. 9a, we find that an increase in the size of the time window results in a decrease in classification accuracy because a time window which is too large is not good for handling sudden drift. From Fig. 9e, we find an increase in the size of the time window correlates with an increase in classification accuracy, since a small time window cannot correctly represent the weather concept. We conclude that the choice of time window has a significant impact on the performance of CM-DRIFT.

Compared with the other two distance functions, Euclidean outperforms on all the datasets. Therefore, we recommend using Euclidean as default option for most situations.

D. Evaluating CM-DRIFT with different base learners

Our method filters the data sample to represent the current concept more accurately. Further, it is not restricted to any particular base learner for classification tasks. In this section, we present an experiment designed to evaluate the performance

TABLE IV: The classification accuracy of the drift region-based data sample filtering method with different base learners.

	SEA	RH	Usenet 1	Usenet 2	NOAA Weather	Spam
kNN	0.8623	0.8520	0.7157	0.7271	0.7906	0.9507
NB	0.8748	0.8706	0.7307	0.7907	0.7014	0.9479
Tree	0.8573	0.7842	0.6979	0.7693	0.7542	0.9287
SVM	0.8748	0.9077	0.7343	0.7757	0.7794	0.9643

TABLE V: Statistical test based on the classification accuracy in Table IV

Post-hoc test after Conover	NB	Tree	SVM
$\lvert R_i - R_{\text{kNN}}\rvert$	2	6	7
p-value	0.245761	0.025767	0.01305

of CM-DRIFT with different base learners. The parameters settings remain as per Sections IV-A and IV-B.

Experiment 10. Varying the base learner. CM-DRIFT is not limited on kNN. It can embed other base learners as well. To evaluate how different base learners affect accuracy, we also show the result of CM-DRIFT with three commonly used base learners other than kNN: Naive Bayes, Deicision Tree and SVM. The base learner is retrained with the data sample D_t that had been updated by CM-DRIFT.

The results of six datasets are shown in TABLE IV. This experiment revealed clear discrepancies based on the different characteristics of each base learner and dataset. For example, the RH dataset appears more suited to SVM since SVM is good at finding a separating hyperplane between data of two classes. The proposed data sample filtering method with kNN (Column 6) returned the highest accuracy on the imbalanced dataset NOAA Weather and is the right choice for imbalanced data given the comparative performance with the other three.

TABLE V shows the statistical test result of the comparison between different base learners where notations are similar to them in TABLE III. It can be seen that CM-DRIFT could achieve significantly greater accuracy when the base learner is Tree or SVM on the tested data streams. Therefore, we suggest trying different base learners when CM-DRIFT is applied to other data streams. In summary, any data-driven base learner could be applied to optimize accuracy, which should be considered when processing data in different scenarios.

E. Computation efficiency analysis

In this section, we analyze the computation efficiency for each algorithm. For concise descriptions, we use the following notations: d is the dimension, $w = |w_p| = |w_c|$, $n = |DriftSet|$, and $N = |\mathcal{X}|$.

Computation efficiency analysis for Algorithm 1 (RNR). Line 2-7 are a kNN search process. Therefore, the complexity is $d \log w$. Line 8 is to order the result of Line 2-7, which is $w \log w$. However, we implement this step with Line 2-7. Therefore, it is actually 1. For the while loop in Line 9-33, if every $Checklist$ goes into this loop, the complexity is $(1+2+\cdots+w) = w(1+w)/2$. The complexity for RNR is at most $d \log w + \frac{w^2}{2} + \frac{w}{2}$.

Computation efficiency analysis for Algorithm 2 (CM-DRDI). Line 1-2 activate RNR. Therefore, the complexity will not exceed $2 \times (d \log w + \frac{w^2}{2} + \frac{w}{2})$. Line 3 computes the distance between w_p and w_c. Therefore, the complexity is $(w+w)^2 = 4w^2$. Line 3-7 involve $\arg\min$, the complexity being at most $2w$. Line 8-13 can be directly computed given the results from Line 1-7. The complexity for CM-DRDI is at most $2d\log w + w^2 + w + 4w^2 + 2w = 2d\log w + 5w^2 + 3w$.

Computation efficiency analysis for Algorithm 3 (CM-DRIFT). Line 1 activates RNR, so the complexity is at most $d\log w + \frac{w^2}{2} + \frac{w}{2}$. Given $DriftFlag$ is always true: the complexity for Line 3-8 is N; Line 9-16 are a kNN search process with complexity of $d\log n$; Line 17-21 will activate CM-DRDI, which is no greater than $2d\log w + w^2 + w + 4w^2 + 2w = 2d\log w + 5w^2 + 3w$. Line 23-31 evolve $W_{new'}$ checks, that reaches a maximum of w. The complexity for CM-DRIFT at its greatest is $\max\left\{3d\log w + d\log n + 5.5w^2 + 3.5w + N, w\right\}$, with a greatest complexity of $3d\log w + d\log n + 5.5w^2 + 3.5w + N \sim O(w^2) + O(N)$.

Based on the computation efficiency analysis, CM-DRIFT will have high complexity if the length of data sample $\mathcal{X}$ increases. As shown in Algorithm 3, when drift is not detected and the newly arrived instance is easily classified (i.e., far from the decision boundary), this instance will be stored in the data sample. Therefore, CM-DRIFT might be slow for data streams without the drift problem. As a solution for this situation, we suggest introducing redundancy removal techniques to ensure that the length of the data sample will not exceed a workable extent.

V. Conclusions and further studies

In this paper, we proposed a drift region-based data sample filtering method which can remove obsolete data and add new data for training. The effectiveness of our method is guaranteed by theoretic proof, and validated on 6 datasets that contain different types of drift.

Our future research endeavors will seek to improve the performance of CM-DRIFT on high-dimension data and reduce the complexity by evolving redundancy data instances removal algorithms. In addition, it is possible to further improve the performance if our method is embedded with other incremental algorithms.

Acknowledgment

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

References

[1] B. Krawczyk and A. Cano, "Adaptive ensemble active learning for drifting data stream mining," in *International Joint Conference on Artificial Intelligence*, Macao, China, Aug. 10-16, 2019, pp. 2763–2771.

[2] J. Lu, A. Liu, Y. Song, and G. Zhang, "Data-driven decision support under concept drift in streamed big data," *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 157–163, 2020.

[3] P. Duda, L. Rutkowski, M. Jaworski, and D. Rutkowska, "On the parzen kernel-based probability density function learning procedures over time-varying streaming data with applications to pattern classification," *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1683–1696, April 2020.

[4] A. Liu, J. Lu, and G. Zhang, "Concept drift detection via equal intensity k-means space partitioning," *IEEE Transactions on Cybernetics*, 2020.

[5] X. Tian, W. W. Y. Ng, and H. Wang, "Concept preserving hashing for semantic image retrieval with concept drift," *IEEE Transactions on Cybernetics*, pp. 1–14, 2019.

[6] W. W. Ng, X. Tian, Y. Lv, D. S. Yeung, and W. Pedrycz, "Incremental hashing for semantic image retrieval in nonstationary environments," *IEEE transactions on cybernetics*, vol. 47, no. 11, pp. 3814–3826, 2016.

[7] J. B. Gomes, E. Menasalvas, and P. A. C. Sousa, "Learning recurring concepts from data streams with a context-aware ensemble," in *Proceedings of the 2011 ACM Symposium on Applied Computing*. TaiChung, Taiwan: ACM, 2011, Conference Paper, pp. 994–999.

[8] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

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

[10] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proceedings of the 17th Brazilian Symposium Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer, 2004, Book Section, pp. 286–295.

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

[12] 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.

[13] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the 7th ACM SIGKDD International Conference Knowledge Discovery and Data Mining*. San Francisco, California: ACM, 2001, Conference Paper, pp. 97–106.

[14] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proceedings of the 9th ACM SIGKDD International Conference Knowledge Discovery and Data Mining*. ACM, 2003, Conference Proceedings, pp. 523–528.

[15] J. Gama, R. Fernandes, and R. Rocha, "Decision trees for mining data streams," *Intelligent Data Analysis*, vol. 10, no. 1, pp. 23–45, 2006.

[16] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *Proceedings of the 8th International Symposium Intelligent Data Analysis*. Springer, 2009, Conference Proceedings, pp. 249–260.

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

[18] H. Wang, "Nearest neighbors by neighborhood counting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 942–953, 2006.

[19] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *SAC*. ACM, 2013, pp. 801–806.

[20] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift," in *Proceedings of the 16th International Conference Data Mining*, 2016, Conference Proceedings, pp. 291–300.

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

[22] A. Liu, J. Lu, F. Liu, and G. Zhang, "Accumulating regional density dissimilarity for concept drift detection in data streams," *Pattern Recognition*, vol. 76, no. Supplement C, pp. 256–272, 2018.

[23] F. Dong, G. Zhang, J. Lu, and K. Li, "Fuzzy competence model drift detection for data-driven decision support systems," *Knowledge-Based Systems*, vol. 143, pp. 284–294, 2018.

[24] F. Gu, G. Zhang, J. Lu, and C.-T. Lin, "Concept drift detection based on equal density estimation," in *Proceedings of the 2016 International Joint Conference Neural Networks*. IEEE, 2016, Conference Proceedings, pp. 24–30.

[25] 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.

[26] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, no. Dec, pp. 2755–2790, 2007.

[27] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the 7th ACM SIGKDD International Conference Knowledge Discovery and Data Mining*. 502568: ACM, 2001, Conference Proceedings, pp. 377–382.

[28] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–31, 2011.

[29] X.-C. Yin, K. Huang, and H.-W. Hao, "De2: Dynamic ensemble of ensembles for learning nonstationary data," *Neurocomputing*, vol. 165, pp. 14–22, 2015.

[30] 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.

[31] Y. Sun, K. Tang, Z. Zhu, and X. Yao, “Concept drift adaptation by exploiting historical knowledge,” *IEEE Trans. Neural Netw. Learning Syst.*, vol. 29, no. 10, pp. 4822–4832, 2018.

[32] B. Smyth and M. Keane, “Remembering to forget: a competence preserving deletion policy for case-based reasoning system,” in *Proceedings of the 14th International Joint Conference Artificial Intelligence (Morgan-Kaufmann, 1995)*, 1995, pp. 377–382.

[33] B. Smyth and E. McKenna, “Modelling the competence of case-bases,” in *European Workshop on Advances in Case-Based Reasoning*. Springer, 1998, pp. 208–220.

[34] F. Liu, W. Xu, J. Lu, G. Zhang, A. Gretton, and D. J. Sutherland, “Learning deep kernels for non-parametric two-sample tests,” in *Proceedings of the 37th International Conference on Machine Learning*, Online, 2020.

[35] F. Dong, J. Lu, K. Li, and G. Zhang, “Concept drift region identification via competence-based discrepancy distribution estimation,” in *Proceedings of the 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, Nov 2017, pp. 1–7.

[36] H. Jiang, “Uniform convergence rates for kernel density estimation,” in *International Conference on Machine Learning*, 2017, pp. 1694–1703.

[37] F. Liu, G. Zhang, and J. Lu, “A novel non-parametric two-sample test on imprecise observations,” in *Proceedings of the 2020 IEEE International Conference on Fuzzy Systems*, Online, 2020.

[38] D. W. Aha, D. Kibler, and M. K. Albert, “Instance-based learning algorithms,” *Machine Learning*, vol. 6, no. 1, pp. 37–66, Jan 1991.

[39] M. Salganicoff, “Tolerating concept and sampling shift in lazy learning using prediction error context switching,” in *Lazy learning*. Springer, 1997, pp. 133–155.

[40] I. Katakis, G. Tsoumakas, and I. Vlahavas, “Tracking recurring contexts using ensemble classifiers: an application to email filtering,” *Knowledge and Information Systems*, vol. 22, no. 3, pp. 371–391, 2009.

[41] I. Katakis, G. Tsoumakas, and I. P. Vlahavas, “An ensemble of classifiers for coping with recurring contexts in data streams,” in *Proceedings of the 18th European Conference Artificial Intelligence*, 2008, Conference Proceedings, pp. 763–764.

[42] I. Katakis, G. Tsoumakas, E. Banos, N. Bassiliades, and I. Vlahavas, “An adaptive personalized news dissemination system,” *Journal of Intelligent Information Systems*, vol. 32, no. 2, pp. 191–212, 2008.

[43] A. Liu, J. Lu, and G. Zhang, “Diverse instance-weighting ensemble based on region drift disagreement for concept drift adaptation,” *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[44] H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining concept-drifting data streams using ensemble classifiers,” in *Proceedings of the 9th ACM SIGKDD International Conference Knowledge Discovery and Data Mining*. Washington, D.C.: ACM, 2003, Conference Paper, pp. 226–235.

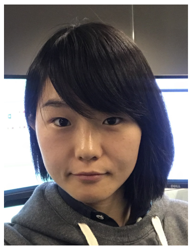
[45] I. Frias-Blanco, J. d. Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, “Online and non-parametric drift detection methods based on hoeffding’s bounds,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2015.

[46] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, “Exponentially weighted moving average charts for detecting concept drift,” *Pattern Recognition Letters*, vol. 33, no. 2, pp. 191–198, 2012.

[47] 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.

Fan Dong is Research Fellow of Australian Artificial Intelligence Institute, University of Technology Sydney. He is the recipient of SIEF STEM+ Business Fellowship. He received the dual Ph.D. degree from University of Technology Sydney and Beijing Institute of Technology in 2018. His research interests include concept drift detection, adaptive learning under concept drift and data stream mining.

Jie Lu (F’18) is a Distinguished Professor and the Director of Australian Artificial Intelligence Institute (AAII) at the University of Technology Sydney (UTS), 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 fuzzy 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 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).



Yiliao Song (M’17) received a M.S. in probability and statistics in mathematics from the School of Mathematics and Statistics, Lanzhou University, China, in 2015. She received the Ph.D. degree in computer science from University of Technology Sydney, Australia. Her research interests include concept drift, data stream mining and real-time prediction. She is a Postdoctoral Researcher at Australian Artificial Intelligence Institute, University of Technology Sydney, Australia.

Feng Liu (M’16) is a Postdoctoral Researcher at Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. He is the recipient of Australian Laureate Postdoctoral Fellowship. He received the Ph.D. degree in computer science from Australian Artificial Intelligence Institute, University of Technology Sydney, Australia, in 2020, and the M.Sc. degree in probability and statistics and the B.Sc. degree in pure mathematics from the School of Mathematics and Statistics, Lanzhou University, China, in 2015 and 2013, respectively. His research interests include domain adaptation and two-sample test. He has served as a senior program committee member for ECAI and program committee members for NeurIPS, ICML, ICLR, AISTATS, ACML, IJCAI, CIKM, FUZZ-IEEE, IJCNN and ISKE. He also serves as reviewers for IEEE-TPAMI, IEEE-TNNLS, IEEE-TFS and IEEE-TCYB. He has received the UTS-FEIT HDR Research Excellence Award (2019), Best Student Paper Award of FUZZ-IEEE (2019) and UTS Research Publication Award (2018).

Guangquan Zhang received the Ph.D. degree in applied mathematics from Curtin University, Australia. He is an Australian Research Council (ARC) QEII fellow and Associate Professor in the Faculty of Engineering and Information Technology at the University of Technology Sydney, Australia. 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 concept drift, fuzzy multi-objective, bilevel, and group decision making, fuzzy measure and fuzzy machine learning. He has published six authored monographs, five edited research books, and over 450 papers including more than 240 refereed journal articles. Dr. Zhang has won nine ARC Discovery Project grants and many other research grants. He has served as a Guest Editor for five special issues of IEEE transactions and other international journals.

APPENDIX A PROOF OF THEOREM 1

This section presents the proof of Theorem 1.

Proof. At first, we define two new variables as following.

$$\underline{K_\Delta}(u) = \sum_{j=1}^{\infty}(k(j\Delta) - k((j+1)\Delta)) \cdot \mathbf{1}\{|u| < j\Delta\}, \qquad (21)$$

$$\overline{K_\Delta}(u) = \sum_{j=1}^{\infty}(k(j\Delta) - k((j+1)\Delta)) \cdot \mathbf{1}\{|u| < (j+1)\Delta\}, \quad (22)$$

where $\Delta > 0$. Thus, it is clear that the following holds for all $\Delta > 0$.

$$\underline{K_\Delta}(u) < K(u) < \overline{K_\Delta}(u). \qquad (23)$$

Then, we assume that the event that Lemma 1 holds, which occurs with the probability at least $1 - 1/m$. And we show the lower bound for $\hat{\mathbb{F}}(X)$. Define

$$\hat{\mathbb{F}}_{\Delta,H}(X) = \frac{1}{\sum_{k=1}^{m}\Omega(r_k)}\sum_{j=1}^{m}\frac{\underline{\mathbb{K}_\Delta}(X, r_j)\cdot\Omega(r_j)}{h_{max}^d}, \qquad (24)$$

where $\underline{\mathbb{K}_\Delta}(X, r_j) = \underline{K_\Delta}\left(\frac{\mathbf{H}_0^{-1/2}(kp(r_j)-X)}{h_{max}}\right)$. It is clear that $\hat{\mathbb{F}}(X) > \hat{\mathbb{F}}_{\Delta,H}(X)$ for all $X \in \mathbb{R}^d$. Let us use the following shorthand $\Delta_{k,j} = k(j\Delta)$. We have

$$\hat{\mathbb{F}}_{\Delta,H}(X) = \frac{\sum_{j=1}^{\infty}(\Delta_{k,j} - \Delta_{k,j+1})\cdot\mathcal{F}_m(B_{\mathbf{H}_0}(X, jh_{max}\Delta))}{md_{max}h_{max}^d}. \quad (25)$$

Next, we have the following property:

$$\mathcal{F}_m(B_{\mathbf{H}_0}(X, jh_{max}\Delta) \geq v_d \cdot (jh_{max}\Delta)^d \cdot F_j, \qquad (26)$$

where $F_j = max\{0, F(X) - \check{u}_X(jh_{max}\Delta/\sqrt{\sigma_d(\mathbf{H}_0)})\}$. Thus, according to Lemma 1, we have

$$\begin{aligned}&\mathcal{F}_m(B_{\mathbf{H}_0}(X, jh_{max}\Delta)\\ &\geq v_dF_j(jh_{max}\Delta)^d - \beta_m\sqrt{v_d}(jh_{max}\Delta)^{d/2}\sqrt{F_j} - \beta_m^2\\ &\geq v_dF_j(jh_{max}\Delta)^d - \beta_m\sqrt{v_d||F||_\infty}(jh_{max}\Delta)^{d/2} - \beta_m^2.\end{aligned}$$

Therefore,

$$\begin{aligned}md_{max}\hat{\mathbb{F}}_{\Delta,h_{max}}(X) &\geq v_d\sum_{j=0}^{\infty}(\Delta_{k,j} - \Delta_{k,j+1})(j\Delta)^dF(X)\\ &- v_d\sum_{j=0}^{\infty}(\Delta_{k,j} - \Delta_{k,j+1})(j\Delta)^d\check{u}_X\left(\frac{jh_{max}\Delta}{\sqrt{\sigma_d(\mathbf{H}_0)}}\right)\\ &- \frac{\beta_m\sqrt{v_d||F||_\infty}}{h_{max}^{d/2}}\cdot\sum_{j=0}^{\infty}(\Delta_{k,j} - \Delta_{k,j+1})(j\Delta)^{d/2} - \frac{\beta_m^2k(0)}{h_{max}^d}.\end{aligned}$$

For the first term, we have

$$\begin{aligned}\lim_{\Delta\to0}v_d\sum_{j=0}^{\infty}(\Delta_{k,j} - \Delta_{k,j+1})(j\Delta)^d &= v_d\int_0^\infty k(t)t^ddt\\ &= \int_0^\infty K(u)du = 1.\end{aligned}$$

Next, we have

$$\begin{aligned}&\lim_{\Delta\to0}v_d\sum_{j=0}^{\infty}(\Delta_{k,j} - \Delta_{k,j+1})(j\Delta)^d\check{u}_X\left(\frac{jh_{max}\Delta}{\sqrt{\sigma_d(\mathbf{H}_0)}}\right)\\ &= v_d\int_0^\infty k(t)t^d\check{u}_X(th_{max}/\sqrt{\sigma_d(\mathbf{H}_0)})dt < \epsilon.\end{aligned} \qquad (27)$$

Lastly, we have

$$\lim_{\Delta\to0}\sum_{j=0}^{\infty}(\Delta_{k,j} - \Delta_{k,j+1})(j\Delta)^{d/2} = \int_0^\infty k(t)t^{d/2}dt < \infty.$$

Thus, taking $\Delta \to 0$, we know

$$\hat{\mathbb{F}}(X) \geq c_l(F(X) - \epsilon - \frac{\beta_m\sqrt{v_d||F||_\infty}}{h_{max}^{d/2}}\cdot\int_0^\infty k(t)t^{d/2}dt - \frac{\beta_m^2k(0)}{h_{max}^d}).$$

Let us redefine

$$\hat{\mathbb{F}}_{\Delta,H}(X) = \frac{1}{\sum_{k=1}^{m}\Omega(r_k)}\sum_{j=1}^{m}\frac{\overline{\mathbb{K}_\Delta}(X, r_j)\cdot\Omega(r_j)}{h_{min}^d},$$

where

$$\overline{\mathbb{K}_\Delta}(X, r_j) = \overline{K_\Delta}\left(\frac{\mathbf{H}_0^{-1/2}(kp(r_j) - X)}{h_{min}}\right).$$

We can similarly obtain

$$\hat{\mathbb{F}}(X) \leq c_u\left(F(X) + \epsilon + \frac{\beta_m\sqrt{v_d||F||_\infty}}{h_{min}^{d/2}}\cdot\int_0^\infty k(t)t^{d/2}dt + \frac{\beta_m^2k(0)}{h_{min}^d}\right).$$

The result follows. □